AD-A101 127   CALIFORNIA UNIV LOS ANGELES DEPT OF COMPUTER SCIENCE   F/G 12/1
MARKOV MODELS FOR MULTIPLE BUS MULTIPROCESSOR SYSTEMS.(U)
FEB 81   M A MARSAN, M GERLA   N00014-79-C-0866
UNCLASSIFIED   UCLA-ENG-8203   NL

1 OF 1
AD
A101127

UCLA

END
DATE
FILMED
7-81
DTIC

END
DATE
FILMED
7-81
DTIC

UCLA

COMPUTER SCIENCE DEPARTMENT

MARKOV MODELS FOR MULTIPLE BUS MULTIPROCESSOR SYSTEMS

Marco Ajmone Marsan
Mario Gerla

February 1981
Report No. CSD 810304

COMPUTER SCIENCE DEPARTMENT OFFICERS

Dr. Gerald Estrin, Chairperson
Dr. Bertram Bussell, Vice Chairperson
Mrs. Arlene C. Weber, Management Services Officer
Room 3731 Boelter Hall

COMPUTER SCIENCE DEPARTMENT RESEARCH LABORATORY

Dr. Leonard Kleinrock, Head
Room 3732 Boelter Hall

This report is part of a continuing series of technical reports initiated in January 1981 and is issued by the Computer Science Department Research Laboratory at UCLA. This technical report presents the latest research results established by the department members and its research staff. It is directed to the professional community and ranges from the presentation of short technical contributions to complete Ph.D. dissertations. Its function is to make available the latest results of our continuing research in a timely fashion. For a complete list of reports in this series you may contact The Department Archivist, at the address below.

University of California, Los Angeles
Computer Science Department
School of Engineering and Applied Science
3732 Boelter Hall
Los Angeles, California 90024

UCLA-ENG-8203 12

University of California

Los Angeles

Markov Models For Multiple Bus Multiprocessor Systems.

by

Marco Ajmone Marsan

Mario Gerla

PREFACE

The research described in this report, "Markov Models For Multiple Bus Multiprocessor Systems," UCLA-ENG-8203, by Marco Ajmone Marsan and Mario Gerla, was carried out as part of the Research in Distributed Processing, sponsored by the Office of Naval Research, Contract No. N00014-79-C-0866 under the direction of A. Avizienis, Principal Investigator, B. Bussell, M. Ercegovac, M. Gerla, S. Parker and D. Rennels, Co-Principal Investigators, in the Computer Science Department, School of Engineering and Applied Science, University of California, Los Angeles.

MARKOV MODELS FOR
MULTIPLE BUS MULTIPROCESSOR SYSTEMS

Marco Ajmone Marsan and Mario Gerla

UCLA Computer Science Department
University of California, Los Angeles

ABSTRACT - Markovian models are developed for the performance analysis of multiprocessor systems intercommunicating via a set of busses. The performance index is the average number of active processors, called processing power. From processing power a variety of other performance measures can be derived as dictated by the specific processor application. Exact models are first introduced, and are illustrated with a simple example. The computational complexity of the exact models is shown to increase very rapidly with system size, thus making the exact analysis impractical even for medium size systems. To overcome the complexity of computation, several approximate models are introduced. The approximate results are compared with the exact ones and found tc be surprisingly accurate for a wide range of configurations. Simulation is used to validate the analytic models and to test their robustness.

This research was supported in part by the Office of Naval Research under contract N00014-79-C-0866 and in part by a NATO grant.

M. Ajmone Marsan is currently on leave from Politecnico di Torino, Istituto di Elettronica e Telecomunicazioni, Torino, Italy.

1. INTRODUCTION

Tightly connected multiprocessor systems are characterized by the presence of several processing units and one or more common memory areas, used by the processors for the exchange of information and, possibly, the storage of common code and data structures of non frequent use. Processors and common memories are connected by some kind of communication system, usually called interconnection network.

Early multiprocessor systems were developed using crossbar networks to connect processors and memories. A widely known crossbar multiprocessor system is C.mmp, the Carnegie Mellon multiminicomputer [WULF72]. The performance of crossbar multiprocessors has been widely analyzed in recent years [BHAN75, BASK76, HOOG77, SETH77, WILL78].

With the availability of inexpensive microprocessors, multiprocessor systems with a very large number of components are now becoming feasible and cost effective. For such systems a crossbar interconnection network may be intolerably expensive and in general it would provide a bandwidth much higher than needed. A more attractive alternative is represented by bus-oriented interconnection networks. Single or multiple bus architectures can be used, according to the bandwidth required for the specific application. These interconnection networks are generally called "multiple-bus" or "highway deficient" [WILL78] networks. Some papers addressing the analysis of bus systems appeared very recently in the literature [HOEN77, FUNG78, WILL78].

This report presents exact and approximate Markovian models for the analysis of multiple-bus multiprocessor systems. Section 2 describes the basic multiprocessor system investigated in this study. In section 3 the model for performance analysis is presented and the assumptions on system operations are discussed. Section 4 derives a variety of application-oriented performance indices . Section 5 provides an exact model for a simple crossbar architecture. Section 6 discusses exact models for general multibus architectures, whereas section 7 derives some approximate, but computationally very efficient models. In section 8 stochastic Petri net models are introduced. In section 9 exact and approximate analytic results are compared, and simulation results are presented.

2. THE MULTIPLE PROCESSOR SYSTEM

This study considers multiple processor systems that exchange information through a common memory which consists of several modules. Processors and common memory modules are connected by a set of "global busses". Each global bus can connect any processor to any memory module. Every processor is also connected (and has exclusive access) to a private memory. The block diagram of a system with 3 processors, 3 memory modules and 2 busses is shown in fig. 1.

The exchange of information is accomplished by first writing the information in the appropriate common memory module and then reading it from the destination processor. Due to the sharing of both memory modules and busses, contention may arise, causing processors to queue for a resource which is currently in use. If the number of busses b is greater or equal to the smaller between the number of processors p and the number of memories m, i.e. $b \geq \min(m,p)$, then the contention is only caused by the sharing of memory modules. Therefore, a processor can always find a free bus to access a free common memory. If, on the other hand, the inequality is not satisfied, a processor may be forced to wait for a memory which is currently free because no bus is available.

Multiple processor systems for which the inequality holds are usually known as "crossbar" architectures. Note that in general it is not wise to set $b > \min(m,p)$ unless we want to add some redundancy in the interconnection network for reliability purposes. In fact, the availability of extra busses does not affect the crossbar system model, nor does it improve its throughput.

Multiple processor systems for which the first inequality does not hold are usually called "highway deficient" systems or "(multiple) bus" architectures (where the word multiple is dropped in the case of $b=1$). For these systems we assume throughout this report that $p \geq m > b$. The case $m > p$ can be analyzed using the same techniques described here; the models are generally simpler than those presented in this report.

It is possible to construct a queueing network model for the analysis of both types of systems. The general case is shown in fig. 2. Processors join memory queues, and before proceeding to service (i.e. accessing memory) they must be granted a permit (bus). The permit is returned upon completion of service. The general model is thus a closed queueing network with p classes of customers and with passive resources [CHAN78, KELL76b], which in this case represent the busses. In the case of crossbar architectures the presence of busses can be ignored, thus making the analysis substantially simpler than for multiple bus systems.

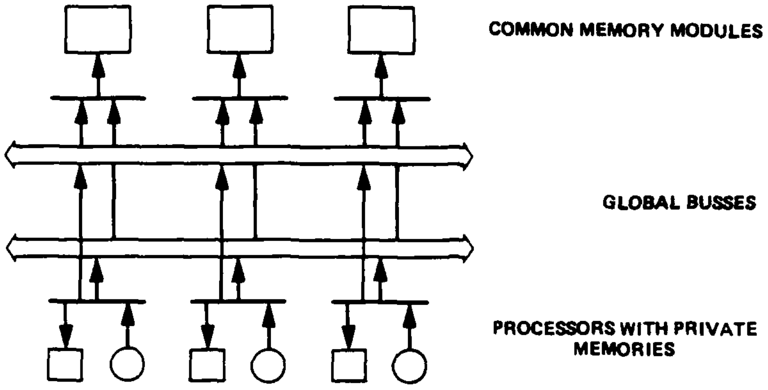


Fig. 1 - Block diagram of a 3x3x2 system.

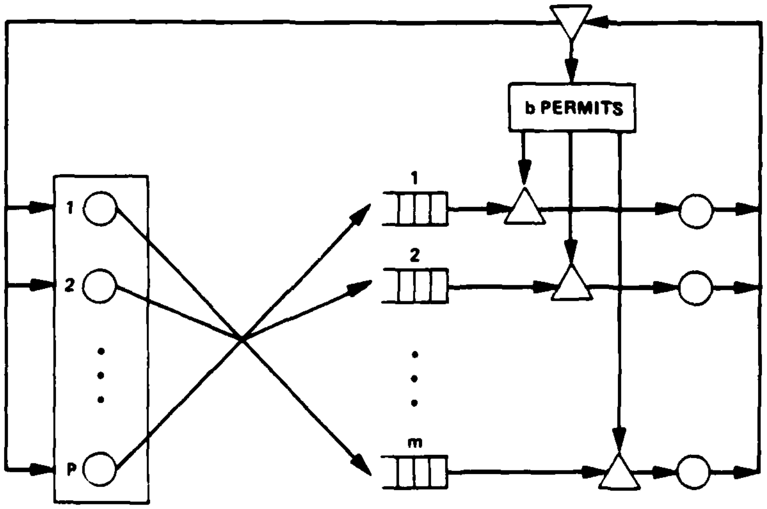


Fig. 2 - Closed queueing network model.

3. THE MODEL

Models of multiprocessor systems are developed both to gain a deeper understanding of their behavior and to obtain a set of performance indices that can be used to guide the design of actual systems.

A model cannot include all the details of the system, rather, it is an abstraction of the real system including the features relevant to the analysis. Different models are generally constructed, depending on the nature of the application and the degree of detail required by the study. In our case, the central feature of the system is the overall processing capability limitation due to the contention for memories and busses. Our models therefore will focus on the loss of processing power due to this contention.

In general we say that a processor can be in one of three different states:

(1) The processor can execute in its private memory.

(2) The processor can exchange data with other cooperating processors, by reading from, or writing into the common memory modules.

(3) The processor can be waiting to access a common memory module.

We say that a processor is ACTIVE when it is in state (1), and the goal of our analysis is to determine the average percentage of time for which processors are active.

By introducing an ergodic assumption we can say that the above quantity is equal to the average number of active processors divided by the total number of processors. Such quantity is usually known in the literature as Processing Efficiency. As the number of processors is a known constant we can simply evaluate the average number of active processors, called Processing Power of the system (P).

$$P = E\,[\#\text{ active processors}] \tag{1}$$

P is the main performance index considered in the sequel. Other important performance measures are simply related to P, as shown in section 4.

The following assumptions are made regarding the operation of the system:

a) Processors perform a background activity that only requires accesses to the processor's private memory.

b) From time to time processors exchange information, and thus access the common memory, performing read/write operations.

c) The duration of the access to the common memory is an independent, exponentially distributed random variable with mean $1/\mu_j$ for the j-th memory module.

d) When a proc:ssor requires access to a common memory module, a path is immediately established (with zero delay) between the processor and the referenced memory module, provided that a bus is available and the memory is not being accessed by another processor.

e) If a path cannot be established the processor idles, waiting for the necessary resource(s) (This may not be true for multiprocessor systems using an interrupt mechanism. The hypothesis is conservative anyway).

f) Upon memory access completion, memory and bus are immediately released (with zero delay) and the processor resumes its background activity. The interval between subsequent access requests, is an independent, exponentially distributed, random variable with mean 1 / λ_j for the j-th processor.

g) An access request from processor i is directed to memory j with probability p_{ij} . Thus, the access rate from processor i to memory j is defined as $\lambda_{ij}=\lambda_i p_{ij}$.

The above assumptions guarantee that a Markovian model can be constructed. Unfortunately, this does not guarantee that a solution (closed form or numerical) can then be easily obtained. In particular, such models show an explosion of the number of states when the number of system components is increased. The analysis becomes rapidly very tedious even for moderately complex systems.

In order to reduce the number of states we introduce three further assumptions.

h) All processors are assumed to have equal common memory access rate, λ, and all memories are assumed to be equal, so that the average memory access time is the same for all memories and all processors ($1/\mu$).

i) A uniform reference model is assumed; this implies that every access request from every processor is directed to any memory with equal probability 1/m, where m is the number of common memory modules.

l) When a bus goes idle, the next processor to use the bus is selected at random among the heads of the queues referencing memories which have become free.

In formulae, assumptions h) and i) state that:

$$
\begin{aligned}
&\lambda_1 = \lambda_2 = \cdots = \lambda_p = \lambda \\
&\mu_1 = \mu_2 = \cdots = \mu_m = \mu \qquad\qquad (2) \\
&p_{ij} = \frac{1}{m} \qquad \text{all } i,j \\
&\lambda_{ij} = \frac{\lambda}{m} \qquad \text{all } i,j
\end{aligned}
$$

With these additional assumptions we succeed in performing an exact analysis of some moderately complex systems, but still cannot attack very large problems.

The equal processor access rate assumption in h) was shown to be a conservative one in the single bus case [AJMO80] and is expected so also in the more general case of multiple busses.

Processors access the common memory modules to perform either read or write operations; we do not distinguish between the two operations in our models, and do not therefore account for the fact that a processor may attempt to read data which is not present in common memory. This results in the processor going idle, with consequent throughput reduction. This feature can be included in the markovian model, but the state space is greatly expanded. A more system oriented approach can be pursued, by assuming that a fraction α of the accesses is for write operations and a fraction $(1-\alpha)$ is for read operations. A read operation finds the required information with probability q. Assuming that the access request generation process is not altered by not finding the desired information, the actual time spent in useful computation is decreased by a factor $(1-q)(1-\alpha)$. Thus, the actual processing power of the system is simply obtained by applying the above factor to the computed value of P. Obviously in this case it is necessary to estimate the values of q and α, which depend on many system parameters.

Using all the above assumptions we can now construct a Markov chain to model the behavior of the system.

The state of the Markov chain is defined by the 2p-tuple

$$(m_1, s_1, m_2, s_2, \ldots, m_p, s_p) \tag{3}$$

where:

m_i is the memory referenced by processor i

s_i is the state of processor i

m_i can take values:

0: processor's private memory

k: k-th common memory module

s_i can take values

0: active

j: queueing (j-th in queue) for module m_i

-1: accessing common memory module m_i

This state definition however is not the most convenient from the computational point of view. In fact, using the theory of "Lumpable" Markov chains [KEME60], we may lump equivalent states and obtain a Markov chain of substantially smaller size. The lumping technique is illustrated by an example in section 5.

The state definition and the degree of lumpability of the chain depend on the policy that is used to assign a free bus to a queueing processor. Assumption 1) is the most convenient from the model complexity point of view, but might not be the one that yields the best performance. Modifications of assumption 1) will be briefly discussed in the sequel.

4. PERFORMANCE MEASURES

The processing power is not the most appropriate performance index for some applications. Other parameters could better describe the quality of the system in some cases. Fortunately, however, many different performance indices can be simply derived from the processing power.

Define λ^* to be the rate at which customers cycle through the queueing network. From Little's result we have:

$$\lambda^* = P\,\lambda \tag{4}$$

Applying again Little's result to the entire memory system including queues and servers we find the average customer delay D:

$$D = \frac{p-P}{P\,\lambda} \tag{5}$$

Finally, subtracting from D the average service time $1/\mu$ we have the average queueing time W:

$$W = \frac{p - P\,(1+p)}{P\,\lambda} \tag{6}$$

where $p = \frac{\lambda}{\mu}$.

The average number of queued processors is:

$$N_q = W\,P\,\lambda = p - P(1+p) \tag{7}$$

therefore the average number of processors accessing common memory modules is:

$$N_s = \frac{D-W}{P\lambda} = P\,\rho \tag{8}$$

The average cycle time C is then easily obtained as:

$$C = W + \frac{1}{\lambda} + \frac{1}{\mu} = \frac{p}{P\,\lambda} \tag{9}$$

From the values of average cycle time, average queueing time, average think time and average service time we can now construct many different performance indices, depending on the particular application.

If the processors are simply updating a data base, a reasonable performance measure could be the ratio of the memory access time to the sum of the access time plus the waiting time. Using the above results, this performance index is expressed as follows:

$$I_{db} = \frac{\frac{1}{\mu}}{\frac{1}{\mu} + W} = \frac{P\,\lambda}{p\,\mu - P(\lambda+\mu)} \tag{10}$$

If, on the other hand, our multiprocessor system is a packet switch operating under heavy load conditions, where input processors process packets and write them into a common memory and output processors read them and again process them before queueing them for output, then the "think" time represents the time necessary to process an incoming (outgoing) packet and the service time represents the time necessary to write (read) a packet from an input (output) processor (note that the exponential read/write time corresponds to exponential packet length distribution). The throughput of the packet switch can then be expressed as:

$$I_{ps} = \frac{p}{2C} = \frac{p\lambda}{2} \tag{11}$$

Note that each packet must be processed by an input and an output processor; both operations require one cycle time, and p packets can be processed simultaneously.

Performance indices for other applications can be constructed in a similar way.

5. CROSSBAR ARCHITECTURES

We begin by presenting as an example the simplest non-trivial case, a 2-processor, 2-memory, 2-bus (2x2x2) system. (Note: the even simpler case of a single bus structure is trivial, and can be analyzed using an M/M/1 queue with finite population. Extensions of the single bus system to different processor access rates and general service distributions are found in [AJMO80]).

A px2x2 system is a crossbar multiprocessor and can thus be studied as a closed queueing network with p classes of customers. Due to the assumptions introduced the solution can be obtained by application of the product form solution [BASK75]. We shall nevertheless construct a Markov chain model, as explained before, to provide a first simple example.

The state definition is in the case of a 2x2x2 system

$$(m_1, s_1, m_2, s_2) \tag{12}$$

and the Markov chain that we obtain using assumptions a) through g) is shown in fig. 3a. In this case no lumping is possible. However, if we add assumptions h) through l) the transition rates are modified as shown in fig. 3b.

We can apply the lumping technique to this Markov chain by defining macrostates as follows:

$$\begin{aligned}
(00) &= [(0000)] \\
(-10) &= [(1\text{-}100),(2\text{-}100),(001\text{-}1),(002\text{-}1)] \\
(-11) &= [(1\text{-}111),(2\text{-}121),(111\text{-}1),(212\text{-}1)] \\
(-1\text{-}1) &= [(1\text{-}12\text{-}1),(2\text{-}11\text{-}1)]
\end{aligned} \tag{13}$$

The lumped chain is shown in fig. 4.

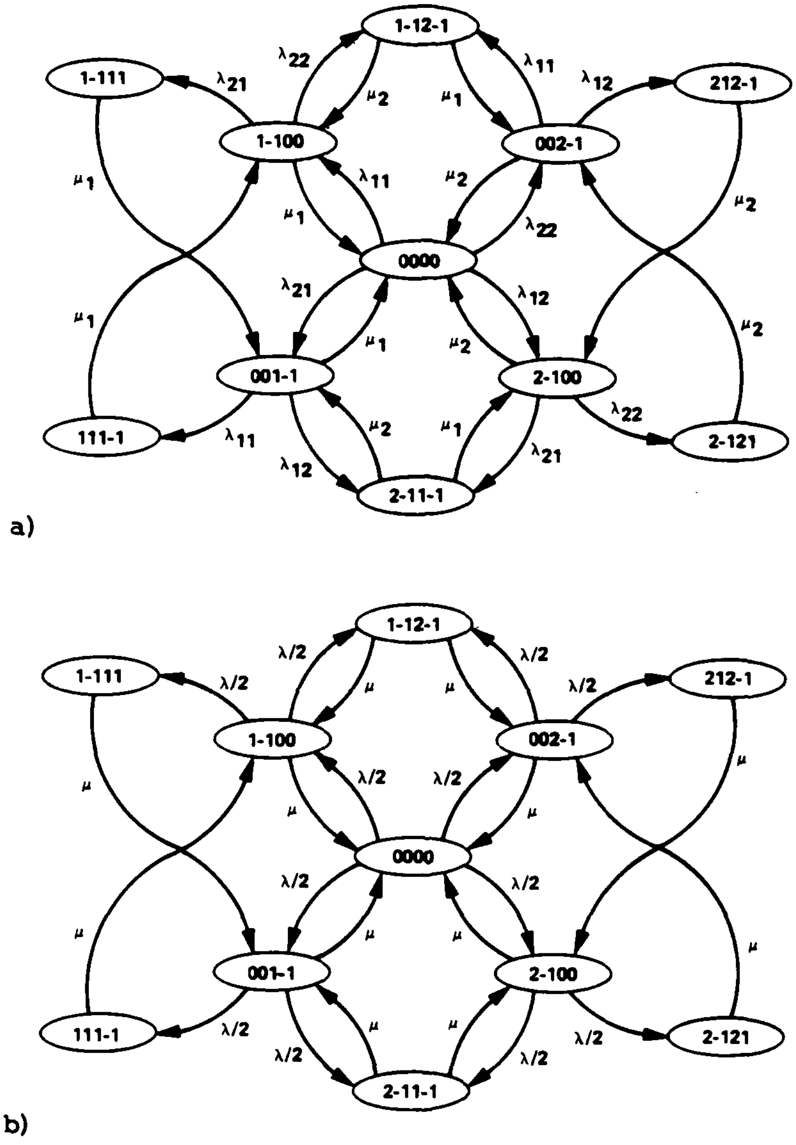


Fig. 3 - Markov chains for the 2x2x2 system:

a - assumptions a through g,

b - assumptions a through l.

Steady state probabilities for the chain in fig. 4 are now very easily evaluated, yielding:

$$\begin{aligned}
&P(-10) = 2p\, P(00) \\
&P(-11) = p^2\, P(00) \\
&P(-1-1) = \frac{1}{2} p^2\, P(00) \qquad (14) \\
&P(00) = [1 + 2p + \frac{3}{2} p^2]^{-1} \\
&p = \frac{\lambda}{\mu}
\end{aligned}$$

The processing power P, defined as the average number of active processors is obtained as:

$$P = 2\ P(00) + P(-10) = 2(1+p)\ [1 + 2p + \frac{3}{2} p^2]^{-1} \qquad (15)$$

As soon as we increase by one the number of processors we realize that the general description is not practical. We have 49 states in this case, that we can lump to 6 macrostates as shown in fig. 5.

The processing power is now obtained as:

$$P = 3\ P(000) + 2\ P(-100) + P(-101) + P(-1-10) \qquad (16)$$

In the same manner we get the lumped chain in the case of four processors that is shown in fig. 6.

In this case we see that in the lumped chain we have two states with two processors accessing the common memory and two processors in queue. State a is such that both processors queue for the same memory and state b is such

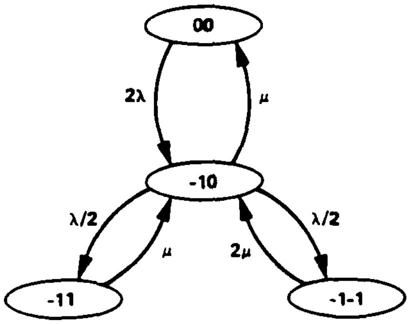


Fig. 4 - Lumped Markov chain for the 2x2x2 system.

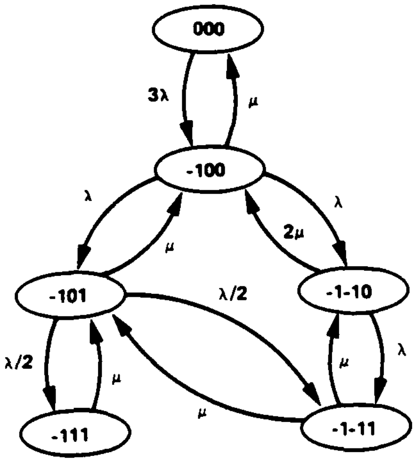


Fig. 5 - Lumped Markov chain for the 3x2x2 system.

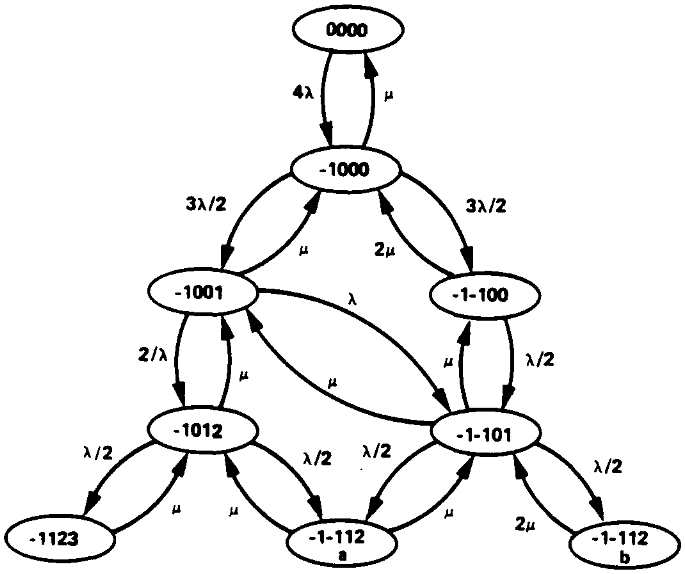


Fig. 6 - Lumped Markov chain for the 4x2x2 system.

that the two processors queue for different memories.

The state definition for the lumped chain in the px2x2 case is:

$$(n_m, n_{q_1}, n_{q_2}) \qquad n_{q_1} \geq n_{q_2} \tag{17}$$

where

n_m is the number of processors currently accessing a common memory

n_{q_1} is the number of processors queueing for the memory with the longest queue

n_{q_2} is the number of processors queueing for the second common memory currently accessed (set to zero if only one memory is used).

In the case of a px2x2 system we are not interested in the policy followed to choose the next processor to be served when a bus becomes available: the only thing that can be done is to pick one of the processsors queueing for the memory that has become available (This is true in general for any crossbar architecture). The fact that we choose the first in the queue is irrelevant for the evaluation of the processing power.

We can now draw the lumped chain in the general case of a px2x2 system (Fig. 7). The number of states of the lumped chain, N, can be evaluated as:

$$N = \begin{cases} \frac{p^2}{4} + p + 1 & \text{p even} \\ \frac{p^2}{4} + p + \frac{3}{4} & \text{p odd} \end{cases} \tag{18}$$

The number of active processors associated to each state is:

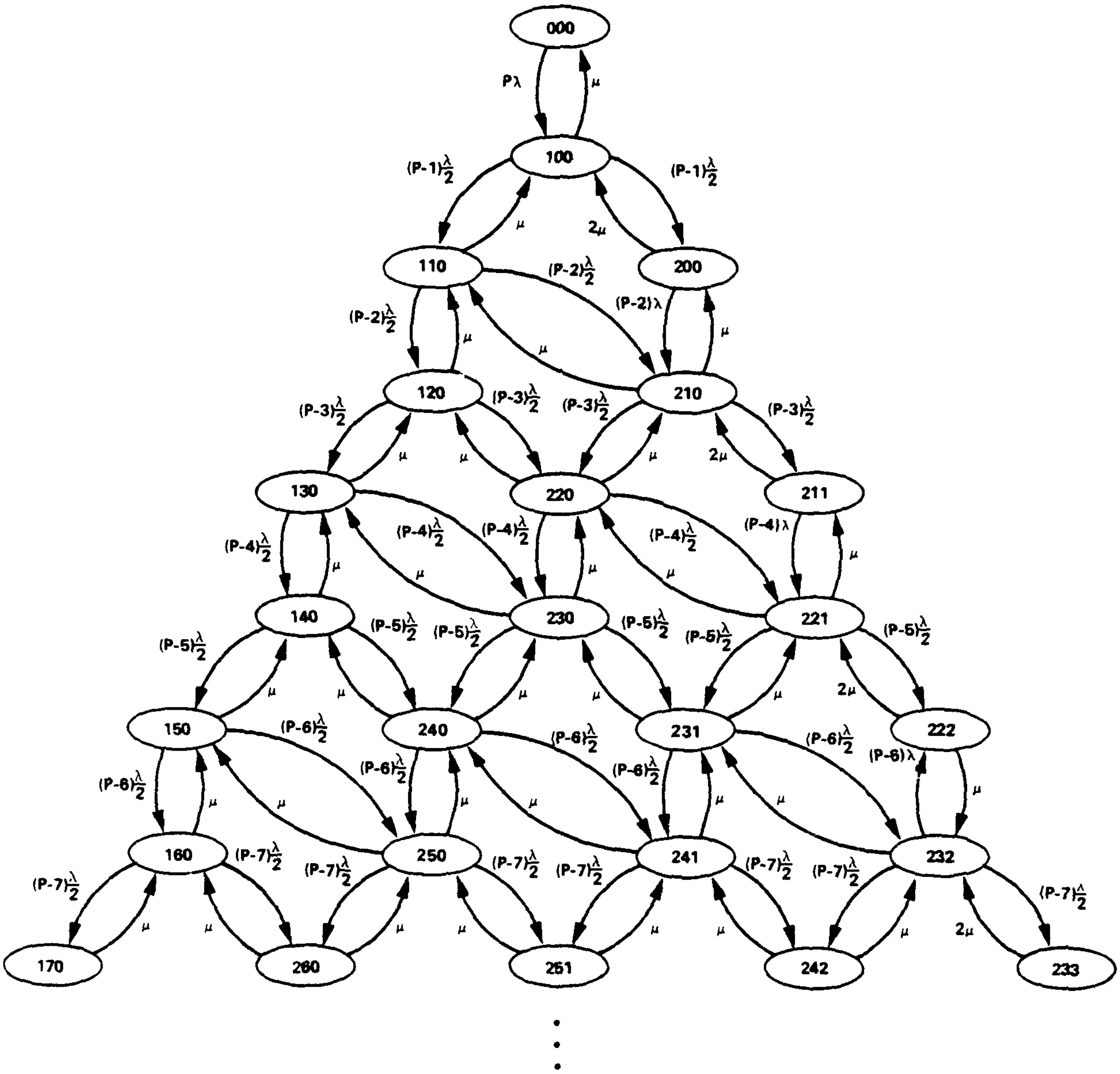


Fig. 7 - Lumped Markov chain for the px2x2 system.

$$p - n_m - n_{q_1} - n_{q_2} \qquad (19)$$

and thus the evaluation of the processing power is straighforward, once the steady state probabilities associated to the states of the Markov chain are evaluated.

In the case of a 3 memory, 3 bus system the state of the lumped Markov chain is defined as:

$$(n_m, n_{q_1}, n_{q_2}, n_{q_3}) \quad , \quad n_{q_1} \geq n_{q_2} \geq n_{q_3} \qquad (20)$$

where

n_m, n_{q_1}, n_{q_2} are defined as before

n_{q_3} is the number of processors queueing for the third common memory currently accessed.

The lumped chain that we obtain is now shown in fig. 8. The transition rates between the states are not shown, but can be easily evaluated.

In the general case of p processors, m memories and m busses ($p \geq m$) the state of the lumped chain is defined by the (m+1)-tuple

$$(n_m, n_{q_1}, \ldots, n_{q_m}) \quad , \quad n_{q_1} \geq n_{q_2} \geq \ldots \geq n_{q_m} \qquad (21)$$

and the definition of the entries is a straightforward extension of the previous case.

The structure of the Markov chain is the same as in fig. 8 up to level 3, then more states must be considered.

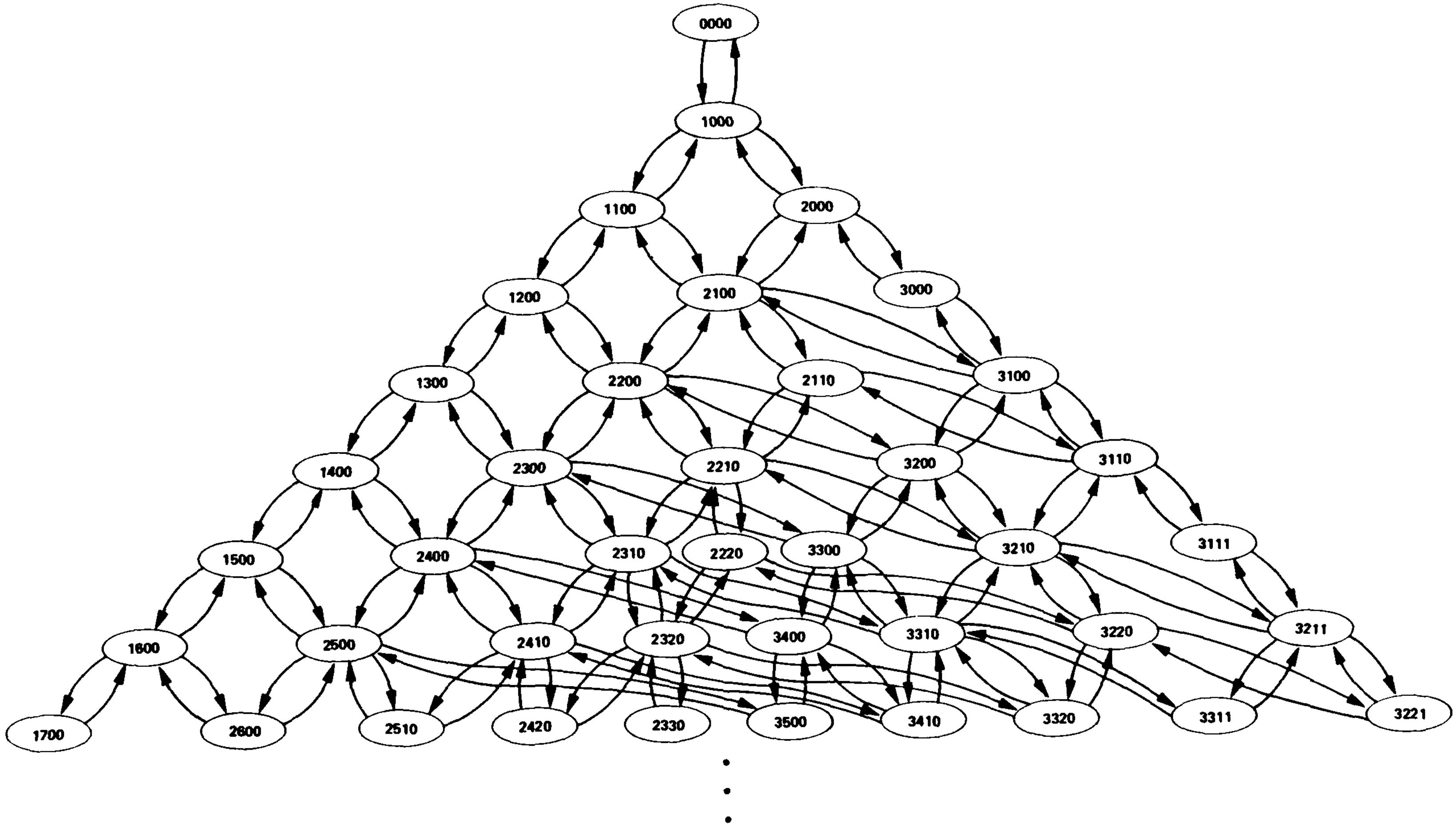


Fig. 8 - Lumped Markov chain for the px3x3 system.

We can express in the general case the transition rates between two states, provided that we specify more precisely the state of the Markov chain.

Given a state as in (21), the entries n_{q_i} must be arranged in decreasing order. We will then have some groups of adjacent entries with the same value.

All the entries of the state can at most increase or decrease by one unit at a time. Only one entry can change at a time.

Given a group of entries n_{q_k}, $n_{q_{k+1}}$, ..., $n_{q_{k+j}}$, all with the same value, only the first entry of the group can increase by one unit, and only the last entry can decrease by one unit. In this manner we are sure to preserve the entries in decreasing order.

Consider now a state

$$(i, q_1, q_2, \ldots, q_m) \tag{22}$$

this state can evolve into at most 2(m+1) other states, which are identified by the following transitions:

$$\begin{array}{ll} i \dashrightarrow i+1 & \\ i \dashrightarrow i-1 & i>0 \\ & \\ q_k \dashrightarrow q_k+1 & q_k \text{ first entry of a group} \\ q_j \dashrightarrow q_j-1 & q_j \text{ last entry of a group, } q_j>0 \end{array} \tag{23}$$

The rates associated to each of these transitions are:

$$R(i \rightarrow i+1) = \begin{cases} p\lambda & i=0 \\ (p-n)(m-i)\dfrac{\lambda}{m} & 0<i<m\ ,\quad n<p \end{cases} \tag{24a}$$

$$R(q_k \rightarrow q_k+1) = \frac{(p-n)}{m}\, l\, \lambda \qquad i<m,\ n<p,\ k\leq i \tag{24b}$$

$$R(i \rightarrow i-1) = (i-s)\,\mu \qquad i-1\geq s \tag{24c}$$

$$R(q_j \rightarrow q_j-1) = l\,\mu \qquad j\leq i \tag{24d}$$

where:

$$n = i + \sum_{k=1}^{i} q_k$$

l = # of entries $q_1, q_2, \ldots, q_i$ that have the same

value of q_j (including q_j itself) (25)

s = # of nonzero entries $q_1, q_2, \ldots, q_i$

The number of active processors, associated to each state is simply p-n; it is thus very easy to obtain the processing power, once we have solved for the steady state probability distribution of the Markov chain.

6. MULTIBUS ARCHITECTURES: EXACT MODELS

For multiple bus architectures, the complexity of the Markov chains is much larger then for crossbar, even when lumping is used. Therefore we can handle only moderately complex systems using the exact state description. For the most general case we must resort to approximate models.

The state definition for the exact lumped chain in the case of a multiple bus system is:

$$(n_m, q_1, q_2, \ldots, q_m) \tag{26}$$

where

n_m is the number of processors currently accessing a common memory

$q_1, \ldots, q_b$ are the numbers of processors queueing for the memories currently accessed, arranged in decreasing order

$q_{b+1}, \ldots, q_m$ are the numbers of processors queueing for a free memory, not accessible because no bus is available, arranged in decreasing order.

Some examples of lumped Markov chains are given in figs. 9 through 13, for 3x3x2, 4x3x2, 5x3x2, 4x4x2 and 4x4x3 systems, respectively.

Note that an increase in the number of processors and/or memories complicates the Markov chain, whereas an increase in the number of busses tends to simplify the Markovian representation. This is due to the fact that the presence of a higher number of busses makes the system more similar to a crossbar, and thus reduces the number of possible queueing situations.

When the number of busses is just one less than the number of processors, the policy for the choice of the next processor to be served is irrelevant. In the other cases the Markov chain depends on such policy. Consider for instance a 4x3x2 system where the next processor served is the one that has been waiting longest. In this case the Markov chain is the one shown in fig. 14, where an asterisk is added to indicate which customer has priority. In general, modifications of assumption 1) require that more information about the state of the system queues is recorded in the Markov chain state description. The resulting chains may thus be much more complex than those obtained using assumption 1).

The general pxmxb case is not easy to handle, even after lumping is applied. We will therefore introduce in the next section some approximations which further reduce the size of the Markov chain and permit us to attack the most general case.

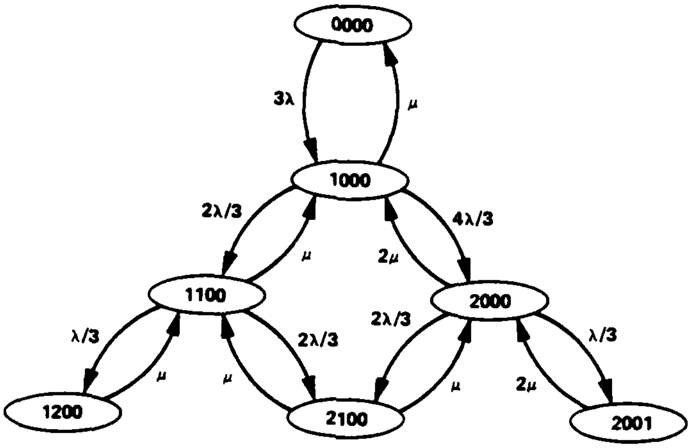


Fig. 9 - Lumped Markov chain for the 3x3x2 system.

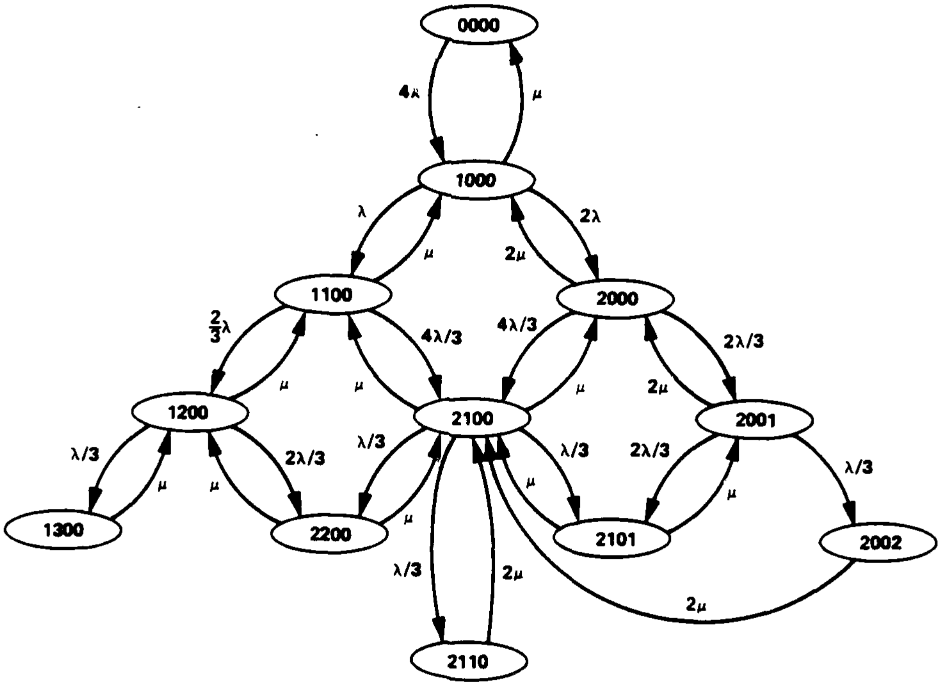


Fig. 10 - Lumped Markov chain for the 4x3x2 system.

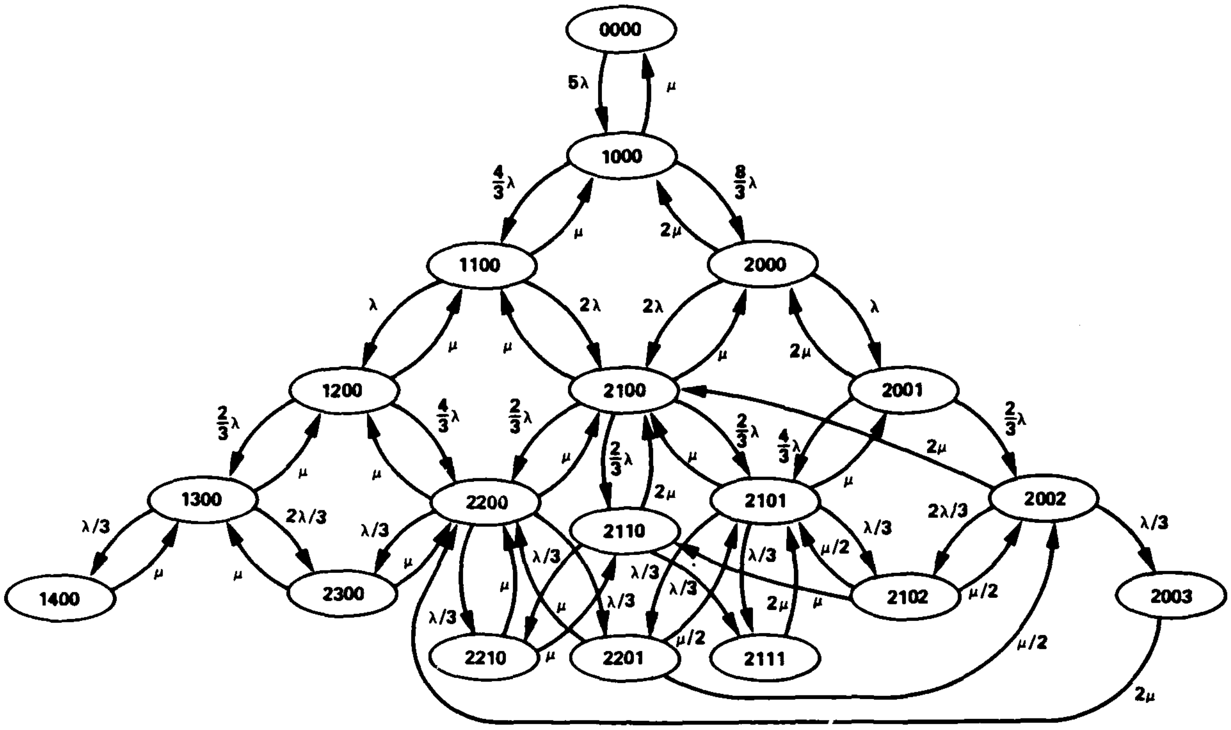


Fig. 11 - Lumped Markov chain for the 5x3x2 system.

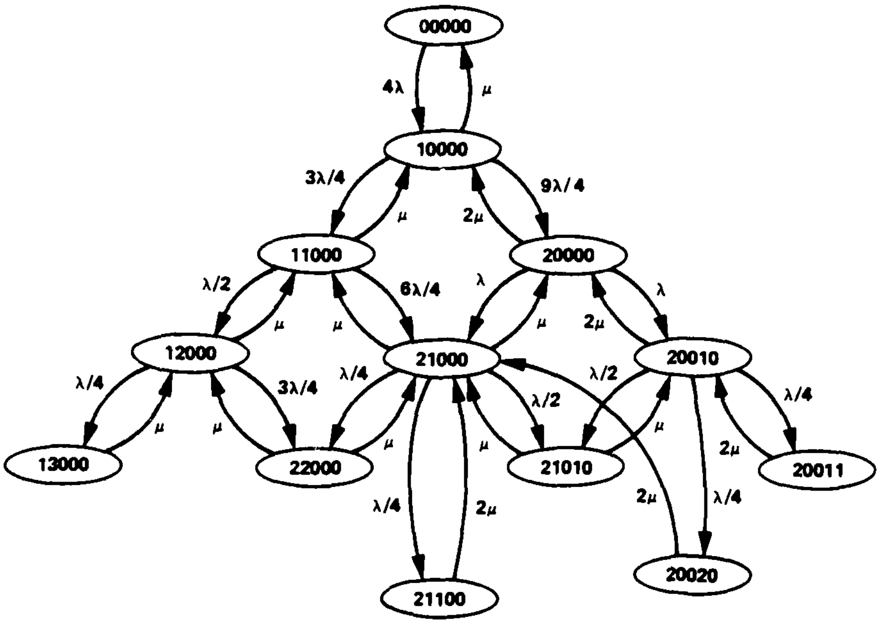


Fig. 12 - Lumped Markov chain for the 4x4x2 system.

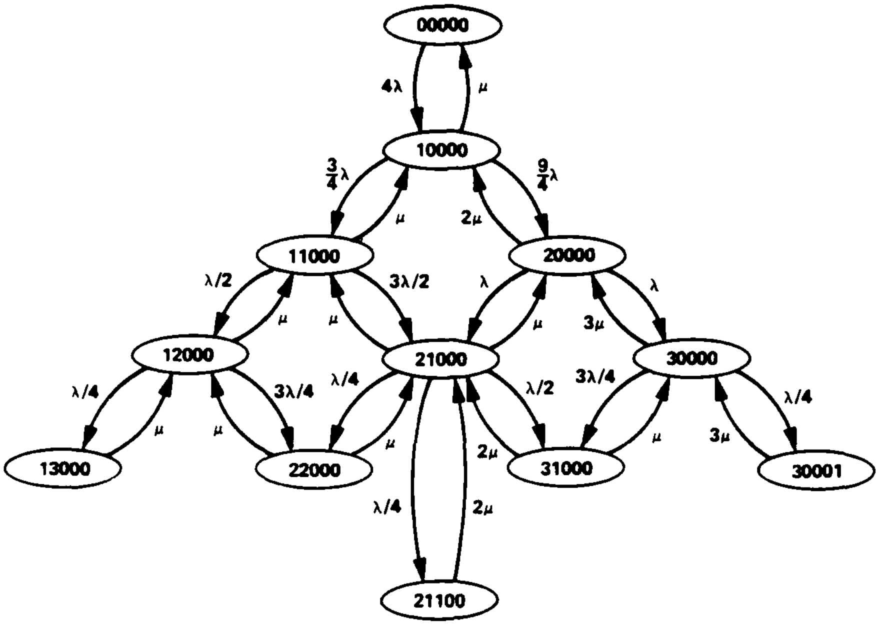


Fig. 13 - Lumped Markov chain for the 4x4x3 system.

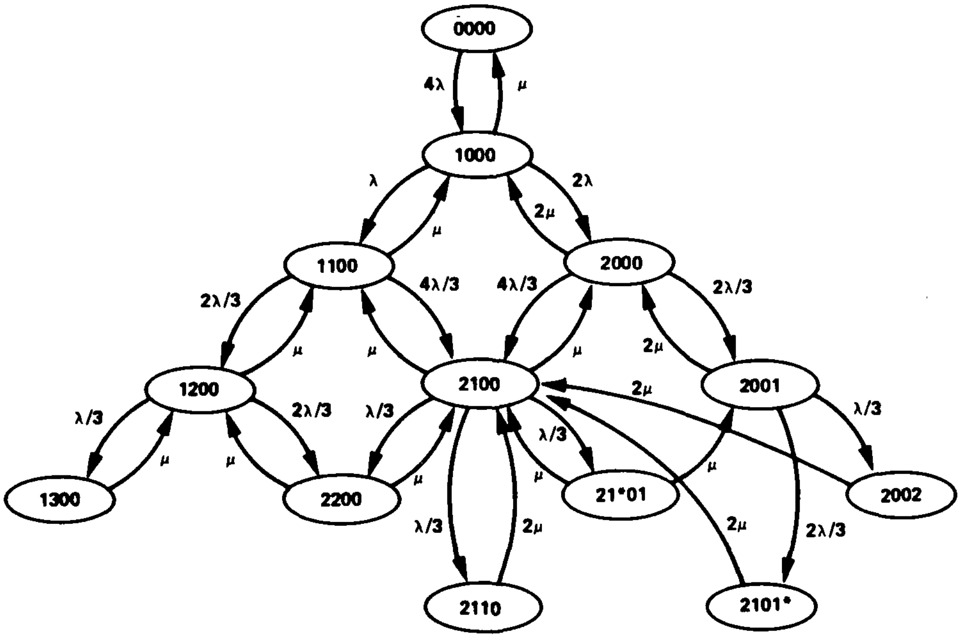


Fig. 14 - Lumped Markov chain for the 4x3x2 system
using a FCFS discipline.

7. MULTIBUS ARCHITECTURES: APPROXIMATE MODELS

The reason for the introduction of approximate Markovian models is that, for general multibus systems, the number of states increases very rapidly with system size. The explosive growth is due to the detailed information that the states must record about the queues inside the system. In particular for each state of the Markov chain the number of customers queued for all common memory modules must be recorded. That is, we not only need to know the number of the queued customers, but also must be concerned with all the possible ways of distributing these customers among the system queues. If we reduce the amount of information about the status of the queues we have no longer a first order Markov chain behavior in the evolution of the system through the state space. The approximate Markov models that we introduce in this section analyze the system behavior by assuming that the transitions between the states with reduced queueing information still satisfy the Markov property. The results that we will obtain in this way are approximate and must then be compared to the exact ones to test their accuracy.

In order to define a simplified model, one needs to specify:

a) the state definition, that is the amount of information used to describe the state of the Markov chain. As was mentioned before we will use reduced information about the queues in the system.

b) the method to calculate the transition rates for the simplified Markov model. As the behavior is approximated by the simplified Markov chain the transition rates must be evaluated according to some empirical rule, and several different rules can be envisioned.

Three different state definitions (named A, B and C) and two heuristic methods for the evaluation of the transition rates (named 1 and 2) were considered. The approximate models are named using the letter referring to the state description and the number referring to the evaluation of the transition rates.

Let us first begin with a very simple model:

Model A1 - The state of the system is simply represented by the total number of processors waiting either for a busy memory or for a busy bus, and by the number of processors currently accessing a common memory module. We thus have a pair

$$(n_m, n_q) \tag{27}$$

where

n_m = # of processors in service

n_q = # of processors queued

The transition rates are evaluated by assuming that each active processor can request any memory module with the same probability (uniform reference model). Furthermore, each queued processor is assumed to request, with uniform probability, any of the common memory modules currently not accessible (this approximation implies that a queued processor can randomly reselect a new memory when a memory or bus becomes unblocked).

If we apply this approximation to the 2x2x2 system and to the 3x2x2 system we find again the exact (lumped) chains. In other words, the above assumptions are automatically verified in such small systems, and therefore no approximation is introduced.

Consider now a 4x2x2 system: in this case we have two states in which two processors are queued. Our approximate chain will consider these two states as a single one. Note, however, that the merging violates the conditions for lumping. Some error will, therefore, appear in the results due to such "prohibited" lumping. The chain that we get is shown in fig. 15.

This approximation can be extended very easily to the px2x2 system, and the resulting chain is shown in fig. 16. The number of states N is in this case only twice the number of processors.

To illustrate the rate computation, consider states (2,p-2) and (1,p-2). The rate from (2,p-2) to (1,p-2) is evaluated by multiplying the rate out of state (2,p-2), which is 2μ, by the probability that none of the p-2 queueing processors is referencing the memory that becomes free. Such probability is $(1/2)^{p-2}$.

Carrying out the analysis for the most general case, we find that the pxmxb system is represented by a Markov chain with b vertical chains (see fig. 17) and a total number of states N, where:

$$N = 1 + b\,[p + \frac{1}{2}\,(1-b)] \qquad (28)$$

A simple upper bound on N is $N \leq p\,b + 1$.

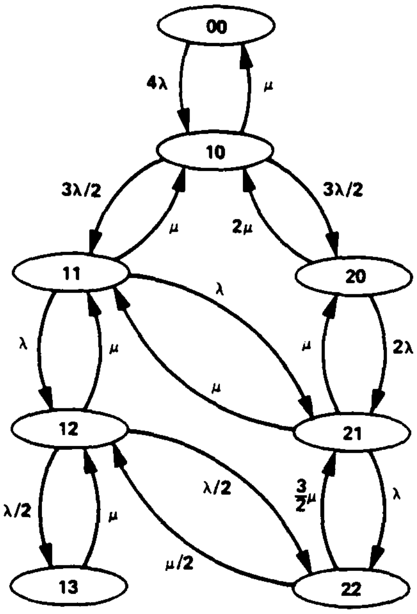


Fig. 15 - Chain of the 4x2x2 system with the approximate model A1.

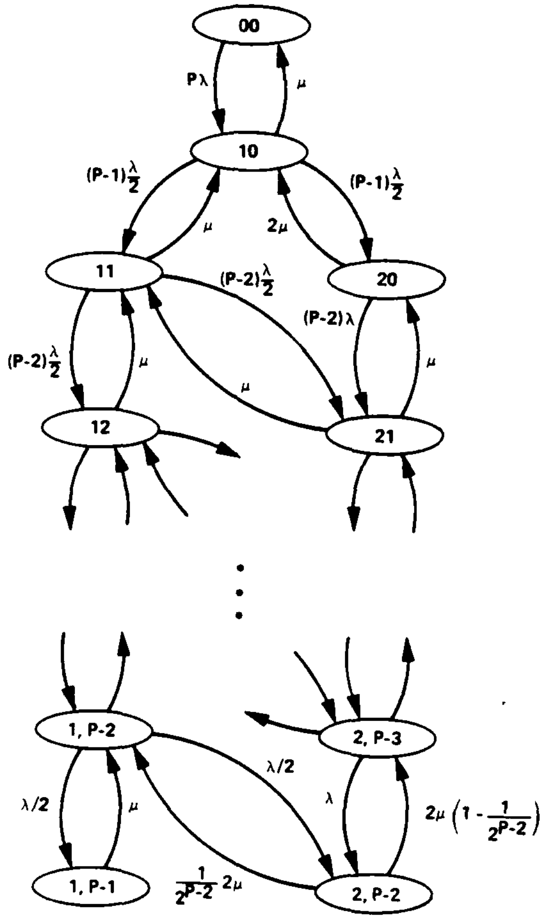


Fig. 16 - Chain of the px2x2 system with the approximate model A1.

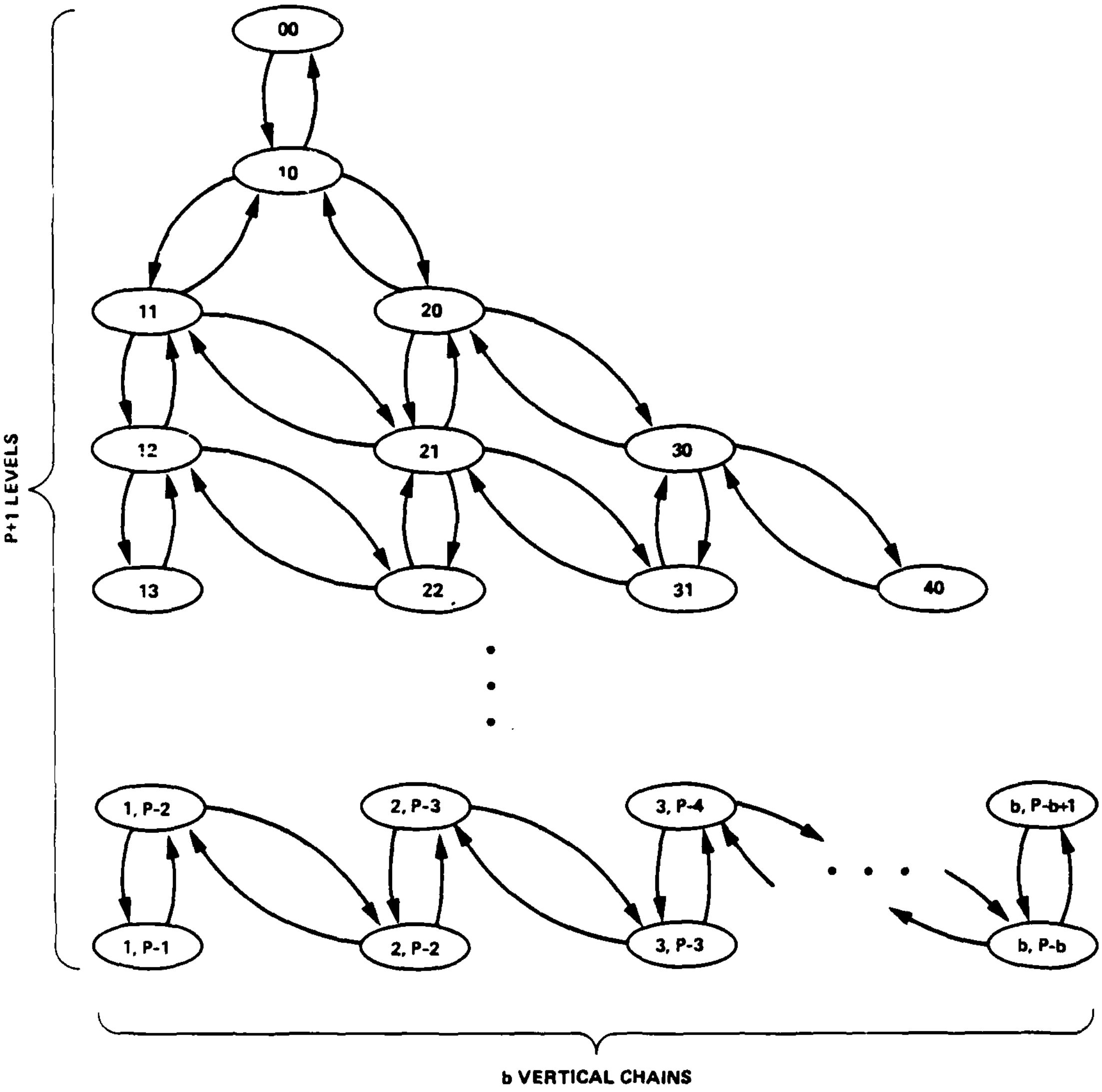


Fig. 17 - Chain of the pxmxb system with the approximate model A1.

The transition rates can be explicitly written for the most general case. Their derivation is reported in appendix 1. Since the number of active processors is p-i-j, the processing power can be simply evaluated once the steady state distribution of the Markov chain is known.

Next we introduce a modification of model A1, by specifying a different method for the calculation of the transition rates:

> Model A2 - The state of the system is defined as in model A1). The transition rates are evaluated using an "averaging" technique.

We describe the model A2 using a 3x3x2 system as an example.

The exact lumped chain for the 3x3x2 system is shown in fig. 9. Using our approximation, the states (2100) and (2001) are merged into state (2,1), even if this violates the lumping conditions. In the approximate chain all *the transition rates are unchanged, except* for those in and out of state (2,1). Namely, +he rates into state (2,1) are obtained by adding the rates into the two merged states. The rates out of state (2,1) are obtained by noting that the total rate out must be 2μ, and that the rate out of the two merged states is μ towards state (1,1) and μ+2μ towards state (2,0). We thus average the rate out of state (2,1), keeping the same ratio. The resulting chain is shown in fig. 18.

Note that no error is made in the approximation if the merged states have equal steady state probability. Otherwise the resulting chain only approximates the exact one.

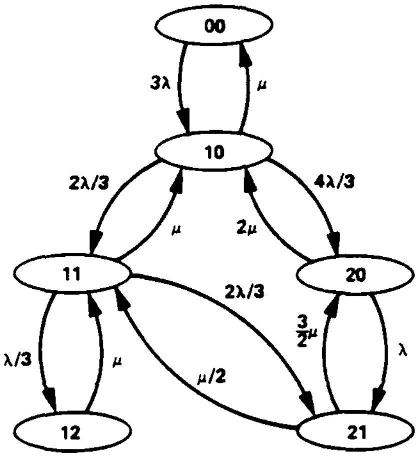


Fig. 18 - Chain of the 3x3x2 system with the approximate model A2.

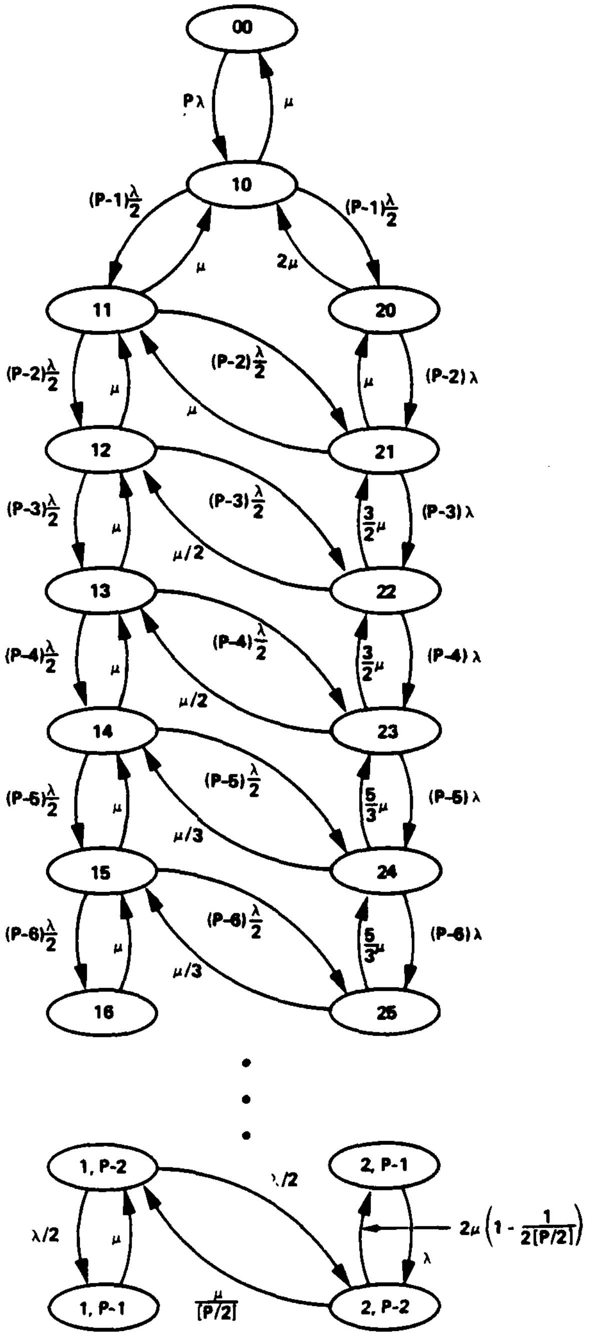
The px2x2 system, using this approximation, is represented by the chain of fig. 19.

The more general case of p processors, m-memories, 2 busses can still be handled, provided that we solve the combinatorial problem of counting the number of states at each level of the exact lumped chain. The level of the state is defined as the sum of the number of processors accessing common memory and the number of queued processors. There is only one state at levels 0 and 1, and there are two states at level 2. For levels larger than two we have one state with $n_m=1$ and n(m,2,k) states with $n_m=2$. The expression of n(m,2,k) is derived in appendix 2. The approximate chain in the case of a pxmx2 system is shown in fig. 20.

The extension to the general pxmxb system with an arbitrary number of busses, requires the counting of the states at each level of a more complex Markov chain, and the corresponding evaluation of new transition rates.

We now consider another definition of system state (yet retaining the rate computation rule of model A2):

> Model B2 - The state of the system is represented by the following triplet: (1) the number of processors accessing a common memory module; (2) the total number of processors waiting either for a busy memory or for a busy bus; and (3) a flag which is set to zero when no processor is queued for a bus, and is set to one when one or more processors are queued for a bus in order to access a free common memory module.



[P/2] = INTEGER ≤ $\frac{P}{2}$

Fig. 19 - Chain of the px2x2 system with the approximate model A2.

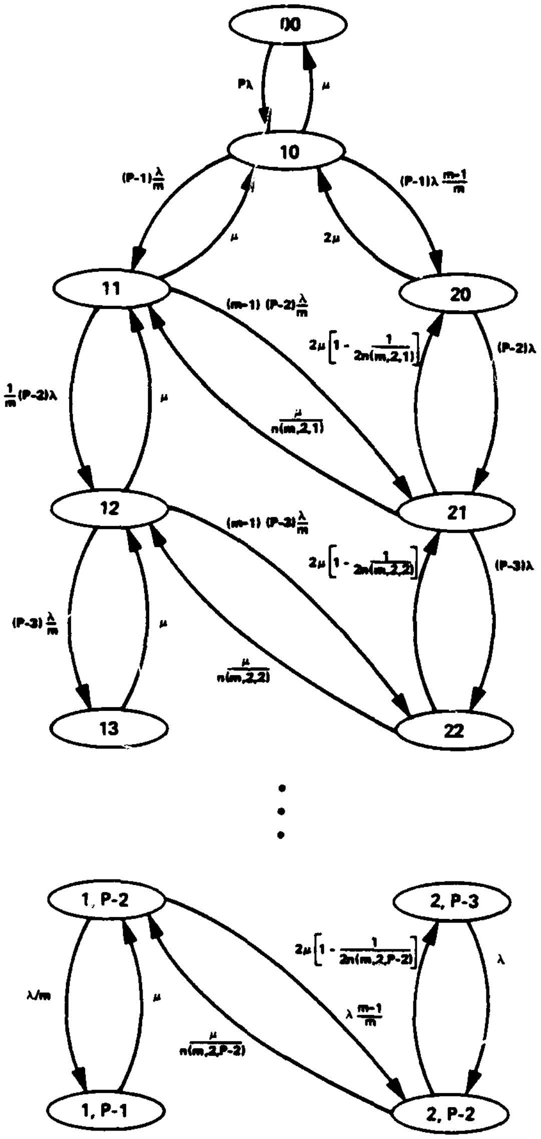


Fig. 20 - Chain of the pxmx2 system with the approximate model A2.

Namely, the state is defined by the triplet

$$(n_m, n_q, f) \tag{29}$$

where

n_m = # of processors accessing common memory

n_q = # of queueing processors

f flag: 0 no queue for a bus

1 one or more processors are queued for a bus

The transition rates are evaluated using the averaging technique described in the approximation A2.

Clearly, model B2 is a refinement of A2, since the state is improved by adding a binary information concerning the system queues.

We immediately recognize that for crossbar architectures the B2 approximation is the same as the A2 approximation, since the flag is always zero (no wait for a bus).

Consider now a 4x3x2 system: the approximate chain is shown in fig. 21. If we compare thi- chain with the exact lumped chain of fig. 10, we see that four states have been merged into two, violating the lumpability conditions. The new transition rates are computed using the averaging technique. The approximate Markov chain for a 5x3x2 system is shown in fig. 22.

The general pxmx2 case can be managed by using the combinatorial results of appendix 2. The resulting chain is shown in fig. 23. The total number of states is in this case

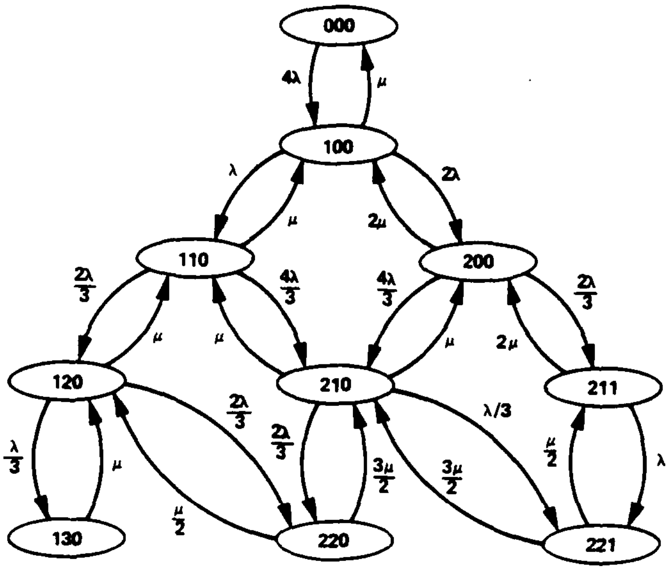


Fig. 21 - Chain of the 4x3x2 system with the approximate model B2.

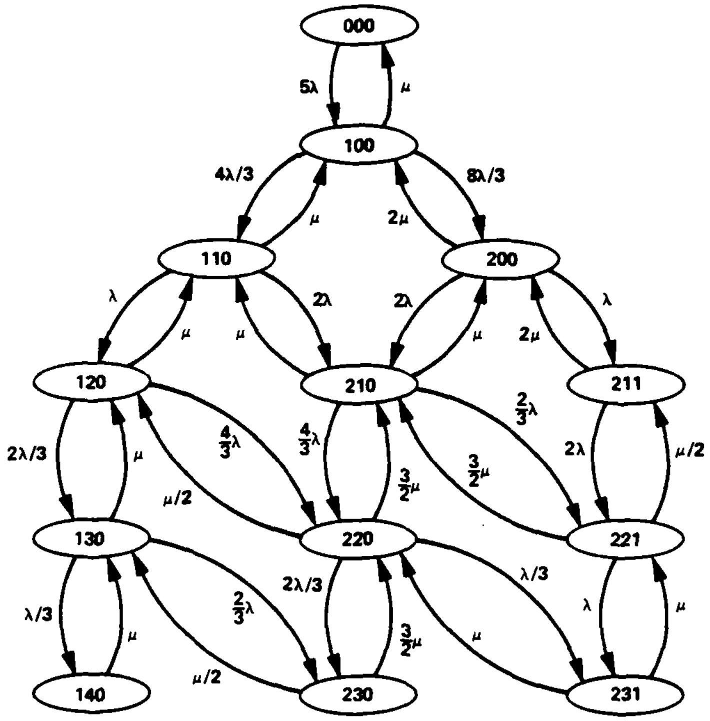


Fig. 22 - Chain of the 5x3x2 system with the approximate model B2.

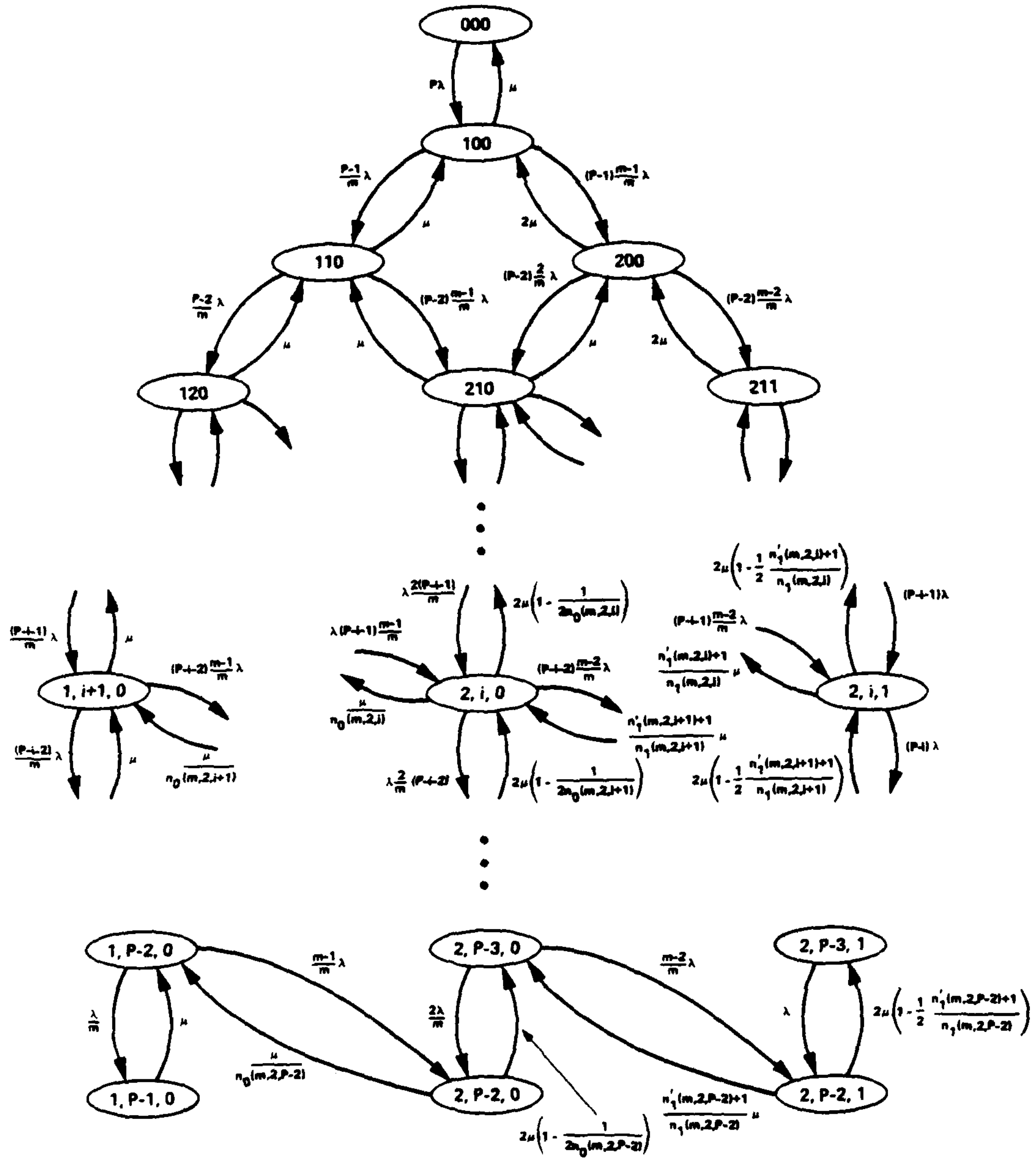

000
$P\lambda$
μ
100
$\frac{P-1}{m}\lambda$
μ
$(P-1)\frac{m-1}{m}\lambda$
2μ
110
200
$\frac{P-2}{m}\lambda$
μ
$(P-2)\frac{m-1}{m}\lambda$
μ
$(P-2)\frac{2}{m}\lambda$
μ
$(P-2)\frac{m-2}{m}\lambda$
2μ
120
210
211
$\lambda\frac{2(P-i-1)}{m}$
$2\mu\left(1-\frac{1}{2n_0(m,2,i)}\right)$
$2\mu\left(1-\frac{1}{2}\frac{n_1'(m,2,i)+1}{n_1(m,2,i)}\right)$
$(P-i-1)\lambda$
$\frac{(P-i-1)}{m}\lambda$
μ
$\lambda(P-i-1)\frac{m-1}{m}$
$(P-i-1)\frac{m-2}{m}\lambda$
$(P-i-2)\frac{m-1}{m}\lambda$
$(P-i-2)\frac{m-2}{m}\lambda$
$\frac{n_1'(m,2,i)+1}{n_1(m,2,i)}\mu$
1, i+1, 0
2, i, 0
2, i, 1
$\frac{\mu}{n_0(m,2,i)}$
$\frac{n_1'(m,2,i+1)+1}{n_1(m,2,i+1)}\mu$
$(P-i)\lambda$
$\frac{(P-i-2)}{m}\lambda$
μ
$\frac{\mu}{n_0(m,2,i+1)}$
$\lambda\frac{2}{m}(P-i-2)$
$2\mu\left(1-\frac{1}{2n_0(m,2,i+1)}\right)$
$2\mu\left(1-\frac{1}{2}\frac{n_1'(m,2,i+1)+1}{n_1(m,2,i+1)}\right)$
1, P-2, 0
$\frac{m-1}{m}\lambda$
2, P-3, 0
$\frac{m-2}{m}\lambda$
2, P-3, 1
$\frac{\lambda}{m}$
μ
$\frac{2\lambda}{m}$
λ
$2\mu\left(1-\frac{1}{2}\frac{n_1'(m,2,P-2)+1}{n_1(m,2,P-2)}\right)$
$\frac{\mu}{n_0(m,2,P-2)}$
1, P-1, 0
2, P-2, 0
2, P-2, 1
$\frac{n_1'(m,2,P-2)+1}{n_1(m,2,P-2)}\mu$
$2\mu\left(1-\frac{1}{2n_0(m,2,P-2)}\right)$

Fig. 23 - Chain of the pxmx2 system with the approximate model B2.

$$N = 3(p-1)+1 \tag{30}$$

As an example the px3x2 chain is shown in fig. 24. In this particular case the combinatorial results can be put in polynomial form (see appendix 2).

The number of active processors associated to each state is $p-n_m-n_q$, thus the processing power can be easily computed, once the steady state probability distribution of the chain is known.

All the preceding approximate models lack of one feature which is very desirable in all analytic models: namely, a closed form solution. We introduce here the simplest possible model, which provides us with a closed form solution.

> Model C2 - The system state is simply the number of active processors: no account is kept of the state of internal queues. The transition rates are evaluated using the averaging technique.

The transition diagram in the case of a pxmx2 system is shown in fig. 25. We have reduced the system description to a birth and death Markov chain, whose solution is easily obtained: denote by $\pi(i)$ the steady state probability of state i, then

$$\pi(i) = \left|\frac{\lambda}{\mu}\right|^{p-i} \frac{p!}{i!} \prod_{k=0}^{p-i-2} \gamma_k^{-1} \, \pi(p)$$

$$\pi(p) = \left| \, 1 + \sum_{j=0}^{p-1} \left| \left|\frac{\lambda}{\mu}\right|^{p-j} \frac{p!}{j!} \prod_{k=0}^{p-j-2} \gamma_k^{-1} \right| \, \right|^{-1} \tag{31}$$

with

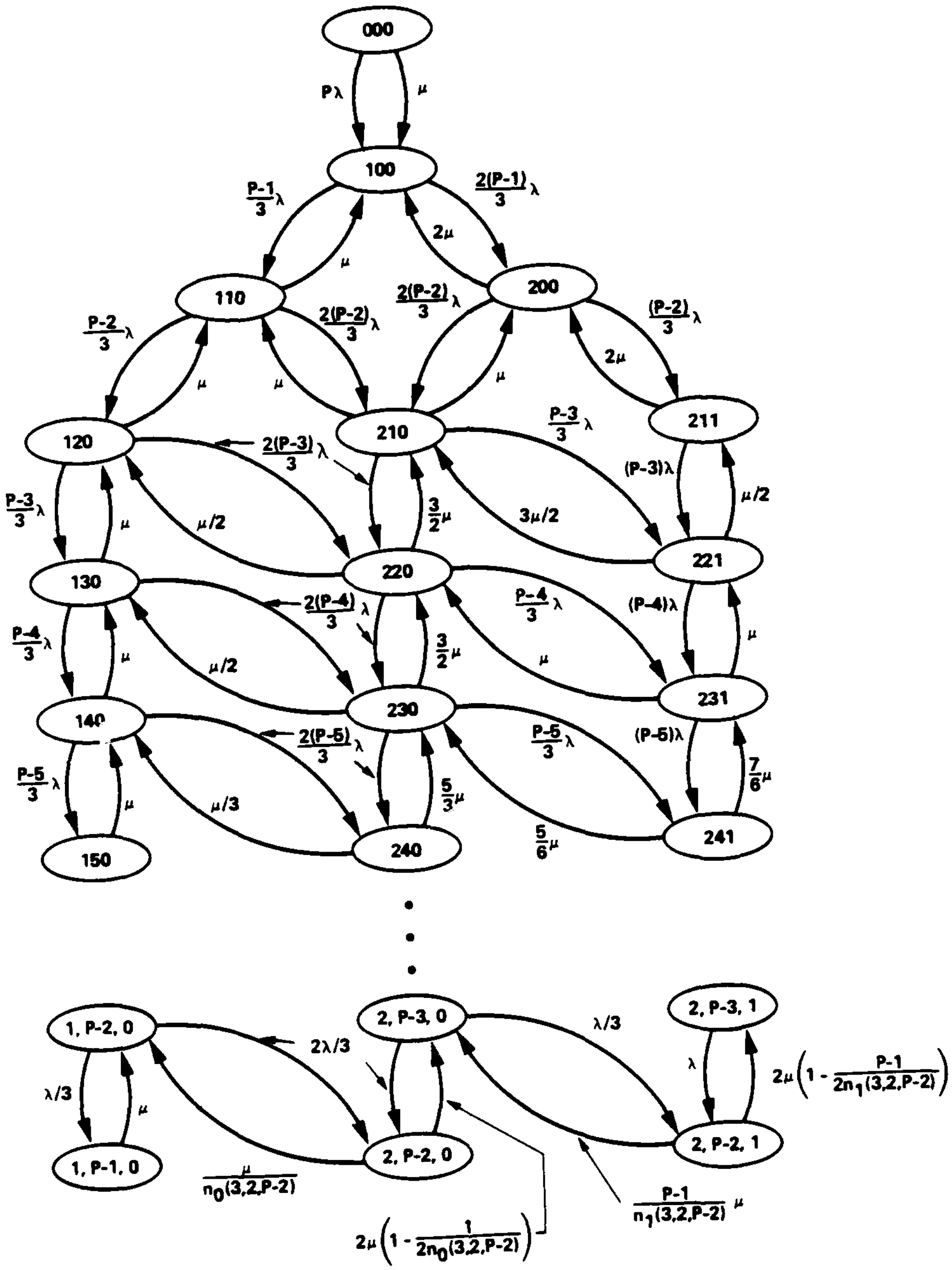


Fig. 24 - Chain of the px3x2 system with the approximate model B2.

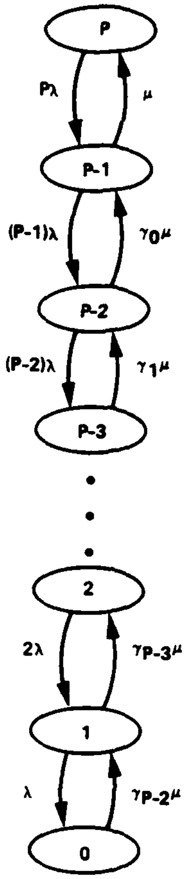


Fig. 25 - Chain of the pxmx2 system with the approximate model C2.

$$\gamma_i = \frac{2\ n(m,2,i)\ +1}{n(m,2,i)\ +\ 1} \tag{32}$$

where n(m,2,i) is defined in appendix 2. The processing power can then be expressed as:

$$P = \sum_{i=1}^{p} \frac{\left|\frac{\lambda}{\mu}\right|^{p-i} \frac{p!}{(i-1)!} \prod_{k=0}^{p-i-2} \gamma_k^{-1}}{1 + \sum_{j=0}^{p-1} \left| \left|\frac{\lambda}{\mu}\right|^{p-j} \frac{p!}{j!} \prod_{k=0}^{p-j-2} \gamma_k^{-1} \right|} \tag{33}$$

The general pxmxb case can also be solved. The resulting Markov chain is shown in fig. 26. The steady state probabilities are in this case:

$$\pi(i) = \left|\frac{\lambda}{\mu}\right|^{p-i} \frac{p!}{i!} \prod_{k=1}^{p-i} \beta_k^{-1}\ \pi(p) \tag{34a}$$

$$\pi(p) = \left| 1 + \sum_{j=0}^{p-1} \left| \left|\frac{\lambda}{\mu}\right|^{p-j} \frac{p!}{j!} \prod_{k=1}^{p-j} \beta_k^{-1} \right| \right|^{-1} \tag{34b}$$

where:

$$\beta_i = \frac{\sum_{j=1}^{b-1} j\ p_j(i) + b \sum_{j=0}^{i-b} \left| p_b(j+b)\ p_{m-b}(i-2b-j+m) \right|}{\sum_{j=1}^{b-1} p_j(i) + \sum_{j=0}^{i-b} \left| p_b(j+b)\ p_{m-b}(i-2b-j+m) \right|}\ ,\ i \geq 1 \tag{35}$$

and $p_i(j)$ is defined in appendix 2.

The expression of the processing power P is then as follows:

$$P = \sum_{i=1}^{p} \frac{\left|\frac{\lambda}{\mu}\right|^{p-i} \frac{p!}{(i-1)!} \prod_{k=1}^{p-i} \beta_k^{-1}}{1 + \sum_{j=0}^{p-1} \left| \left|\frac{\lambda}{\mu}\right|^{p-j} \frac{p!}{j!} \prod_{k=1}^{p-j} \beta_k^{-1} \right|} \tag{36}$$

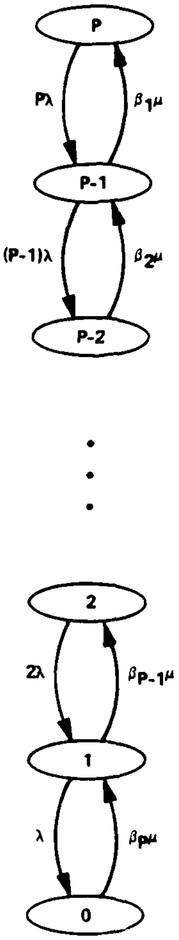


Fig. 26 - Chain of the pxmxb system with the approximate model C2.

8. STOCHASTIC PETRI NET MODELS

Petri net models and derivations thereof [PETR66, HOLT68, HOLT70, PETR73] have been introduced by several authors for the modeling of computer systems [NOE71, NUTT72, NOE73, KELL76b, PETE77, AGER79, SHAP79]. Although in standard Petri nets no measure of time is considered (only a partial ordering of the occurrences of events is established), some of the models presented in the literature allow a measure of the flow of time by introducing the concept of transition times. Transition times are assumed to be deterministic, even in the Random Petri net models introduced by Shapiro [SHAP79]. Molloy [MOLL80] first introduced the idea of random transiton times, by allowing them to be exponentially distributed random variables. We show in this section how such models can be used to describe the behavior of multiple bus multiprocessor systems and to obtain the Markovian models discussed in the previous sections.

For an introduction to Petri nets the reader is referred to the tutorial papers by Peterson and Agerwala [PETE77, AGER79].

Following [AGER79] we define a Petri net (PN) to be represented by a bipartite, directed graph: PN = (T,P,A), where:

$$\begin{aligned} T &= \{t_1,t_2,\ldots,t_n\} && \text{is a set of transitions} \\ P &= \{p_1,p_2,\ldots,p_m\} && \text{is a set of places} \\ A &\leq \{T\times P\}U\{P\times T\} && \text{is a set of directed arcs} \qquad (37) \end{aligned}$$

The set { T U P } forms the set of nodes of the Petri net.

The dynamic properties of the PN can be studied by analyzing the movements of tokens inside the net. A PN with tokens is a marked Petri net MPN = (T,P,A,M). A marking M of a PN assigns tokens to places; M can be viewed as a vector whose i-th component represents the number of tokens assigned to the i-th place p_i. A marking can also be viewed as a mapping from the set of places P to the natural numbers I:

$$M : P \text{ -> } I$$
$$M = \{\nu_1,\nu_2,\ldots,\nu_m\} \qquad (38)$$

It is common practice to represent places by circles, transitions by bars and tokens by black dots. A simple Petri net is shown in fig. 27.

For a given transition t we define the set of input places I(t) as:

$$I(t) = \{\ p \mid (p,t) \in A\ \} \qquad (39)$$

in a similar manner the set of output places is defined as:

$$O(t) = \{\ p \mid (t,p) \in A\ \} \qquad (40)$$

A transition is enabled if the marking M of the Petri net is such that:

$$M(p) > 0 \qquad \text{all } p \in I(t) \qquad (41)$$

Enabled transitions can fire thus removing one token from each input place and putting one token in each output place. The firing of a transition alters the marking of the PN and may then enable other transitions. The dynamic behavior of the Petri net can thus be investigated studying the sequences of markings produced by firing the transitions.

Standard Petri net models do not consider time as a parameter of the net; the firing of a transition is assumed to be instantaneous. Modified models (see for instance the E-net models [NUTT72, NOE73]) allow the introduction of fixed transition times. With stochastic Petri nets the transition times are assumed to be exponentially distributed random variables (possibly with zero mean, thus accounting for immediate transitions). More precisely, the time that elapses between the enabling and the firing of a transition is an exponentially distributed random variable; the firing time is still assumed to be zero, thus in the case of two conflicting transitions the firing of one disables the other.

A continuous time stochastic Petri net (SPN) is thus an extension of the standard Petri net:

$$SPN = (P,T,A,M,\delta) \qquad (42)$$

where δ is the set of the transition rates associated to each transition:

$$\delta = \{\delta_1,\delta_2,\ldots,\delta_n\} \qquad (43)$$

A discrete time SPN can also be introduced, by considering geometrically distributed transition times [MOLL80].

Petri nets are useful in modeling asynchronous concurrent activities in real systems. We can attach a physical interpretation to markings and transitions: a marking can represent the state of the system and a transition can represent an event which modifies the system state. Consider for example the very simple system of fig. 28: two processors access an external common memory. The behavior of this system can be represented by the PN of fig. 27, by giving the following interpretation to places and transitions:

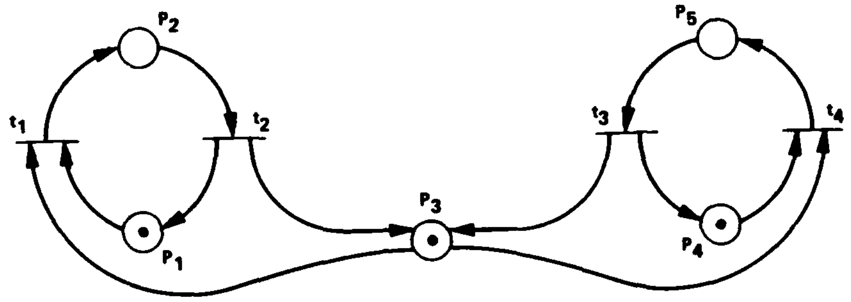


Fig. 27 - A simple Petri net.

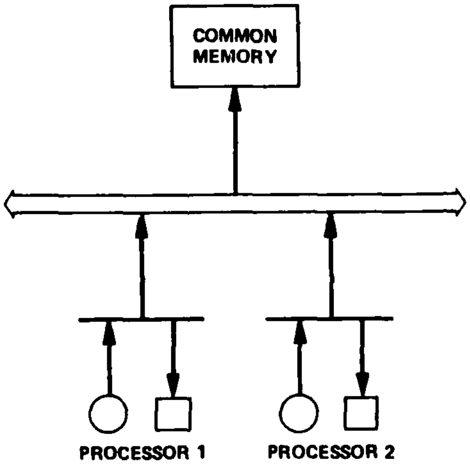


Fig. 28 - Two-processor system.

p_1 processor 1 active

p_2 processor 1 accessing common memory

p_3 bus available

p_4 processor 2 active

p_5 processor 2 accessing common memory

t_1 processor 1 seizes the bus

t_2 processor 1 releases the bus

t_3 processor 2 releases the bus

t_4 processor 2 seizes the bus

With this model we represent the possible conflicts in access requests, but do not explicitly model the queueing of a processor in order to access the common memory. This feature can be obtained by adding two places and two transitions to the net as shown in fig. 29. The interpretation of the added nodes is:

p_6 processor 1 queued

p_7 processor 2 queued

t_5 processor 1 issues a request

t_6 processor 2 issues a request

The marking shown in the figures indicates the initial state of the system. In order to obtain the full definition of the stochastic Petri net we must associate a rate with each transition. Using the same notation as in section 3 we have:

$\delta_1 = \delta_4 = \infty$ immediate transition

$\delta_2 = \delta_3 = \mu$ memory access completion rate

$\delta_5 = \lambda_1$ access rate of processor 1

$\delta_6 = \lambda_2$ access rate of processor 2 (44)

The analysis of Petri net models is usually based on the properties of the reachability set associated to the PN. The Rechability set of a PN is the set of all markings reachable from the initial marking M. A marking M' is immediately reachable from M if it can be obtained from M by firing some enabled transition. A marking M' is reachable from M if it is immediately reachable from M or if it is reachable from any marking immediately reachable from M. The reachability set of the SPN of fig. 29 is easily obtained, and it is shown in fig. 30. Marking 8 is somewhat different from all the others, as it is obtained from markings 2 and by firing a finite rate transition before an immediate transition. Marking 8 is therefore reachable with probability zero.

The number of tokens in any place can be at most one for all markings. This means that the SPN is safe. A place in a PN is said to be safe, if it contains at most one token; if all places of a PN are safe, then the PN is safe. We also note that all transitions are such that for each marking M, there is a marking M', reachable from M in which the transition is enabled. This means that all the transitions in the net are live, hence the PN itself is live. Liveness is an important property as it guarantees that the PN is deadlock-free.

Due to the memoryless property of the negative exponential distribution, the SPN is isomorphic to a continuous time Markov chain as shown by Molloy [MOLL80]. The state space of the Markov chain can be obtained from the reachability set by eliminating those markings that enable an immediate transition ($\delta_i = \infty$). In the case of the SPN of fig. 29 we must eliminate markings 2 and 4 that enable t_1 and t_4, respectively, and marking 8 that enables both. We thus have a 5-state Markov chain that can be represented with the

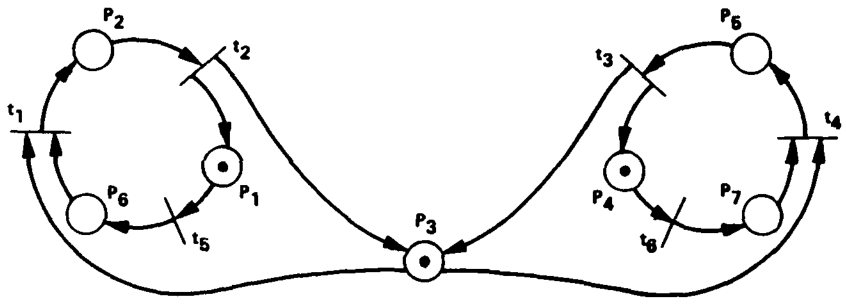


Fig. 29 - Stochastic Petri net model of the two-processor system.

Marking	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1	1	0	1	1	0	0	0
2	0	0	1	1	0	1	0
3	1	0	1	0	0	0	1
4	0	1	0	1	0	0	0
5	0	0	1	0	0	1	1
6	1	0	0	0	1	0	0
7	0	1	0	0	0	0	1
8	0	0	0	0	1	1	0

Fig. 30 - Reachability set of the Stochastic Petri net of fig. 29.

transition diagram of fig. 31, where the state definition is as follows:

$$(s_1, s_2) \tag{45}$$

with:

s_i = state of processor i

w = active

a = accessing common memory

q = queued

The marking that corresponds to the state is also indicated in the figure. The transition rates are those associated with the transition that has to be fired in order to go from one state to the other. In the case of immediate transitions, we consider the state where the immediate transition is enabled to coincide with the state resulting from the firing of the immediate transition.

Consider a 2x2x2 system, as described in section 5. We can represent the behavior of such system using the SPN of fig. 32. The interpretation of places and transitions is a simple extension from fig. 29. The transition rates are:

$$\begin{aligned}
\delta_1 &= \lambda_{11} \\
\delta_2 &= \lambda_{12} \\
\delta_3 &= \delta_4 = \delta_9 = \delta_{10} = \infty \\
\delta_5 &= \delta_{11} = \mu_1 \\
\delta_6 &= \delta_{12} = \mu_2 \\
\delta_7 &= \lambda_{21} \\
\delta_8 &= \lambda_{22}
\end{aligned} \tag{46}$$

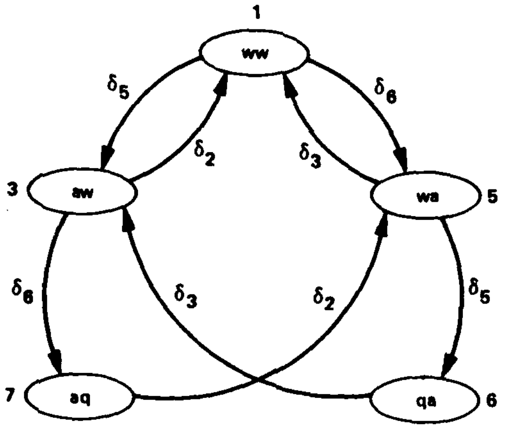


Fig. 31 - Markov chain model of the two-processor system.

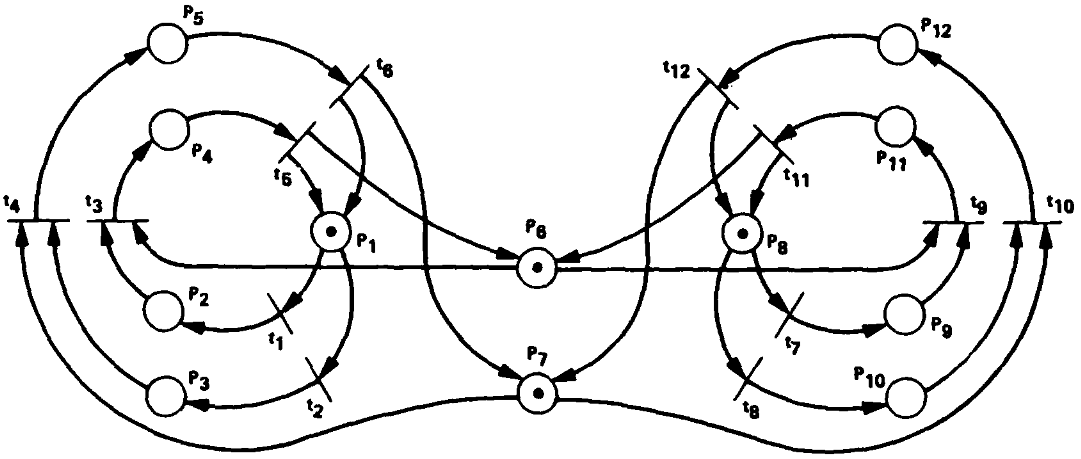


Fig. 32 - Stochastic Petri net model of a 2x2x2 system.

The reachability set is now shown in fig. 33; 23 markings are possible, 4 are reachable with probability zero, 8 of them enable immediate transitions, hence the associated Markov chain has eleven states. The construction of the Markov chain using the rates associated to each transition yields exactly the chain of fig. 3a. From the SPN description of the system we can obtain the Markov chain description presented in the previous sections.

Note that the stochastic Petri net of fig. 32 is safe and live, hence the system (as modeled) is deadlock-free.

Petri nets have been used to describe and model the synchronization of events. In the case of multiprocessor systems that exchange messages through common memories, processors are synchronized in the sense that a message can be read only after it has been written. As we mentioned in section 3 a processor may look for a message in a common memory and not find it. Moreover, the common memory area is limited, it can accomodate only a fixed number of messages (assume that the common memory consists of several buffers which can accomodate one message each). These features of the real system can be included in the SPN model rather easily. Consider again the simple system of fig. 28. The message exchange through finite size memories can be modeled explicitly using the SPN of fig. 34, where the interpretation of places is as follows:

Marking	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
1	1	0	0	0	0	1	1	1	0	0	0	0
2	0	1	0	0	0	1	1	1	0	0	0	0
3	0	0	1	0	0	1	1	1	0	0	0	0
4	1	0	0	0	0	1	1	0	1	0	0	0
5	1	0	0	0	0	1	1	0	0	1	0	0
6	0	0	0	1	0	0	1	1	0	0	0	0
7	0	1	0	0	0	1	1	0	1	0	0	0
8	0	1	0	0	0	1	1	0	0	1	0	0
9	0	0	0	0	1	1	0	1	0	0	0	0
10	0	0	1	0	0	1	1	0	1	0	0	0
11	0	0	1	0	0	1	1	0	0	1	0	0
12	1	0	0	0	0	0	1	0	0	0	1	0
13	1	0	0	0	0	1	0	0	0	0	0	1
14	0	0	0	1	0	0	1	0	1	0	0	0
15	0	0	0	1	0	0	1	0	0	1	0	0
16	0	1	0	0	0	0	1	0	0	0	1	0
17	0	1	0	0	0	1	0	0	0	0	0	1
18	0	0	0	0	1	1	0	0	1	0	0	0
19	0	0	0	0	1	1	0	0	0	1	0	0
20	0	0	1	0	0	0	1	0	0	0	1	0
21	0	0	1	0	0	1	0	0	0	0	0	1
22	0	0	0	1	0	0	0	0	0	0	0	1
23	0	0	0	0	1	0	0	0	0	0	1	0

Fig. 33 - Reachability set of the Stochastic Petri net of fig. 32.

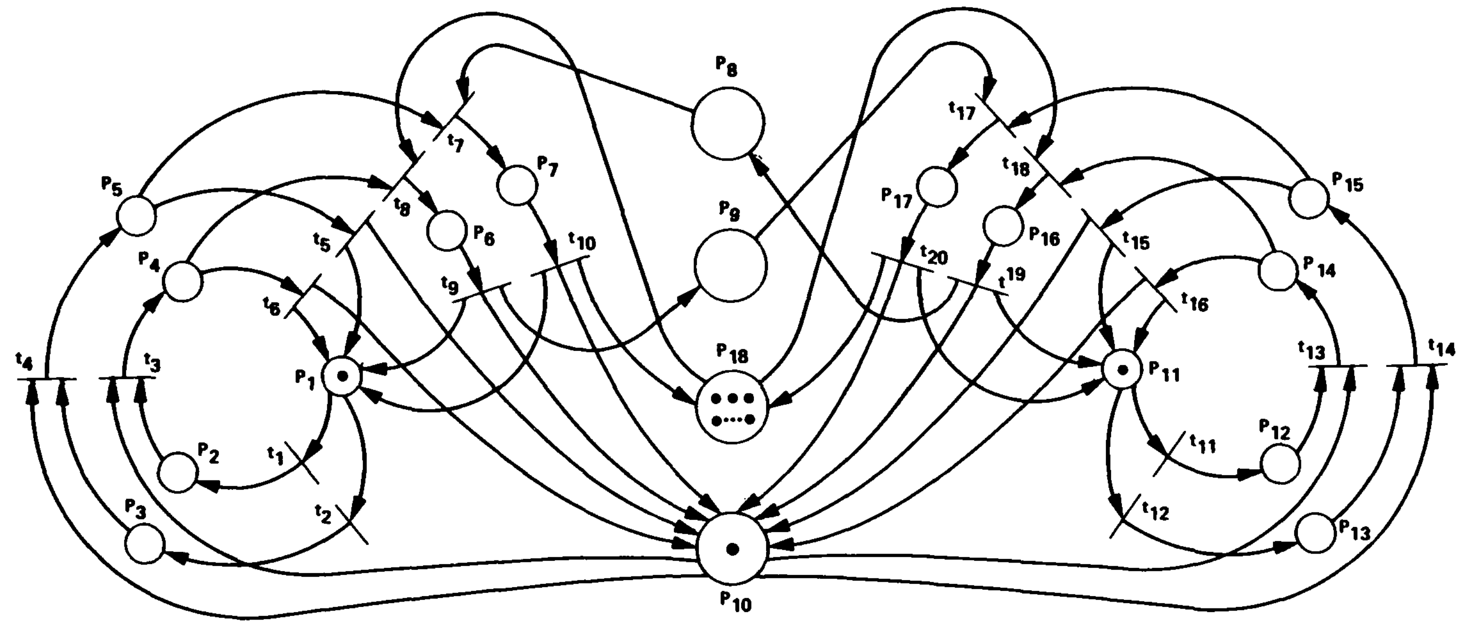


Fig. 34 - Stochastic Petri net model of the two-processor system including synchronization and buffer size.

$p_{1(11)}$ processor 1(2) active

$p_{2(12)}$ processor 1(2) queued for write

$p_{3(13)}$ processor 1(2) queued for read

$p_{4(14)}$ processor 1(2) testing the availability of buffers

$p_{5(15)}$ processor 1(2) testing the presence of messages

$p_{6(16)}$ processor 1(2) writing

$p_{7(17)}$ processor 1(2) reading

$p_{8(9)}$ messages for processor 1(2)

p_{10} bus available

p_{18} buffers in common memory

The interpretation of the transitions is:

$t_{1(11)}$ proc. 1(2) issues a write request

$t_{2(12)}$ proc. 1(2) issues a read request

$t_{3(13)}$ proc. 1(2) seizes the bus for write

$t_{4(14)}$ proc. 1(2) seizes the bus for read

$t_{5(15)}$ proc. 1(2) found no message

$t_{6(16)}$ proc. 1(2) found no buffer

$t_{7(17)}$ proc. 1(2) found a message

$t_{8(18)}$ proc. 1(2) found a buffer

$t_{9(19)}$ proc. 1(2) write ends

$t_{10(20)}$ proc. 1(2) read ends

The immediate transitions in the SPN are:

$$t_3, t_4, t_7, t_8, t_{13}, t_{14}, t_{17}, t_{18} \tag{47}$$

To all other transitions we can assign finite rates, according to the definitions of section 3. The SPN is live, thus the system is deadlock-free, but, in general, it is not safe, as places p_8, p_9 and p_{18} contain more than one token at a time, unless the common memory consists of a single buffer. The SPN is however k-bounded, that is, for each marking the number of tokens in any place of the network is smaller than k, k being the number of buffers available in the common memory. The k-boundedness of the SPN guarantees that the reachability set is finite. Since the size of the state space of the equivalent Markov chain is smaller than or equal to the size of the reachability set, it too is finite.

The Petri net model provides a formal description of the operation of the system: from fig. 34 and the interpretation of places and transitions, we obtain all the information necessary to describe the way the system operates.

The SPN is in this case much more complex than in fig. 29, where we did not explicitly model the synchronization between transmitting and receiving processor, 75 markings are reachable in the single buffer case. Nevertheless, from fig. 34, we can obtain a Markov chain that models the behavior of the system including those features, using the same rules as before. The complexity of the result limits the applicability of these highly detailed models to very small systems.

9. RESULTS

Exact and approximate analytic results were compared by considering a 4x3x2, a 4x4x3 and a 6x4x2 system respectively. The exact chains for the first two systems are shown in fig. 6 and 8, respectively. The exact chain for the third system (not shown here) has 37 states.

The results for the 4x3x2 system are presented in fig. 35. The first column gives the value of $\rho = \frac{\lambda}{\mu}$, the second column shows the exact value of processing power as a function of ρ , evaluated using the exact lumped chain. The other columns show the percentage error which affects the processing power value computed with each of the four approximations introduced in this report. For this case, the exact chain has 12 states, approximations A1 and A2 have 8, approximation B2 has 10 and approximation C2 has 5 states.

The results for the 4x4x3 system are shown in fig. 36, using the same format. The exact chain has again 12 states, whereas the approximate chains have 10, 10, 11 and 5 states, respectively.

In fig. 37 the results for the 6x4x2 system are presented. The number of states are in this case 37, 12, 12, 16 and 7.

A number of observations can be made based on these results. Firstly approximations A1, A2 and B2 seem to yielu upper bounds on the processing power, whereas C2 gives a lower bound. The upper bound can be intuitively explained for approximation A1, since the random redistributing of processors to memories tends to relieve memory congestion and therefore improve performance. The bounds seem to be rather tight, since percentage errors well below 10%

p	exact	A1	A2	B2	C2
.1000e-02	.3996e+01	.00	.00	.00	.00
.1000e-01	.3960e+01	.00	.00	.00	.00
.1000e+00	.3604e+01	.02	.04	.00	-.01
.3000e+00	.2892e+01	.30	.53	.04	-.20
.5000e+00	.2338e+01	.84	1.33	.14	-.50
.1000e+01	.1506e+01	2.15	2.95	.50	-1.12
.3000e+01	.5806e+00	3.82	4.12	1.27	-1.65
.5000e+01	.3559e+00	4.07	4.02	1.49	-1.65
.1000e+02	.1804e+00	4.13	3.79	1.63	-1.58
.1000e+03	.1824e-01	4.06	3.48	1.70	-1.45
.1000e+04	.1826e-02	4.05	3.44	1.70	-1.43

Fig. 35 - Results for the 4x3x2 system.

p	exact	A1	A2	B2	C2
.1000e-02	.3996e+01	.00	.00	.00	.00
.1000e-01	.3960e+01	.00	.00	.00	.00
.1000e+00	.3613e+01	.00	.00	.00	-.18
.3000e+00	.2948e+01	.06	.11	.04	-.82
.5000e+00	.2440e+01	.25	.42	.15	-1.28
.1000e+01	.1651e+01	1.00	1.73	.58	-1.69
.3000e+01	.6847e+00	3.13	5.67	1.78	-1.91
.5000e+01	.4280e+00	3.94	7.30	2.23	-2.09
.1000e+02	.2203e+00	4.63	8.78	2.63	-2.36
.1000e+03	.2258e-01	5.26	10.25	3.01	-2.77
.1000e+04	.2264e-02	5.31	10.40	3.05	-2.83

Fig. 36 - Results for the 4x4x3 system.

p	exact	A1	A2	B2	C2
.1000e-02	.5994e+01	.00	.00	.00	.00
.1000e-01	.5940e+01	.00	.00	.00	.00
.1000e+00	.5386e+01	.07	.06	.01	-.39
.3000e+00	.4167e+01	.89	.59	.15	-2.24
.5000e+00	.3191e+01	1.82	.99	.30	-3.45
.1000e+01	.1858e+01	2.52	.89	.28	-3.73
.3000e+01	.6513e+00	1.83	.27	.07	-2.75
.5000e+01	.3927e+00	1.56	.18	.08	-2.49
.1000e+02	.1970e+00	1.36	.13	.10	-2.29
.1000e+03	.1974e-01	1.19	.12	.12	-2.10
.1000e+04	.1974e-02	1.17	.12	.12	-2.08

Fig. 37 - Results for the 6x4x2 system.

were tipically observed (except for approximation A2 in the 4x4x3 case). Tight upper and lower bounds are extremely useful, as they allow to determine a small range in which the exact result must lie, avoiding the computational complexity of the exact problem.

Secondly, we observe that the largest system (6x4x2) shows the smallest percentage errors. This may be due to the fact that the rate averaging approximation gives better results for higher number of states. If the trend of smaller errors with larger systems were verified for even larger models, then we could conclude that our approximate models are more than adequate for the study of large multibus systems.

In order to study the influence of the simplifying assumptions introduced, and to test the performance of the approximate techniques on larger systems, a simulation program was written in GPSS. Due to the peculiarities of the language, some discrepancies are expected between the simulated systems and the models for which we performed a Markov chain analysis. Nevertheless a comparison between the analytic and the simulation results shows a very good agreement. As an example, in fig. 38 results are shown for the 2x2x2 system.

The influence of the simplifying assumptions was studied taking the 6x4x2 system as a benchmark. Exact and approximate analytic results for this system were shown in fig. 37. First, the impact of memory access time distribution on system performance is tested. Fig. 39 shows the value of the processing power - obtained via simulation - for a 6x4x2 system with fixed memory access time. The fixed access time results are smaller than the exponential access time results in fig. 37, as is expected from known results in queueing theory.

p	analysis	simulation
0.01	1.97	1.98
0.1	1.80	1.81
0.3	1.49	1.46
0.5	1.26	1.26
1.	0.89	0.87
3.	0.41	0.39
5.	0.25	0.24
10.	0.13	0.12

Fig. 38 - Comparison of analytic and simulation results for a 2x2x2 system.

p	P
.001	5.99
.01	5.94
.1	5.42
.333	4.14
.5	3.37
.75	2.49
1.	1.95
3.	0.66
5.	0.40

Fig. 39 - Processing power of a 6x4x2 system with fixed access times.

Next, the uniform memory reference assumption is relaxed by assuming that access requests from any processor are directed to memory 1 with probability α, and are uniformly distributed among all other memories. That is, we set:

$$P_{i1} = \alpha \qquad \text{all } i$$

$$P_{ij} = \frac{1 - \alpha}{m - 1} \qquad \text{all } i,\ j=2,\ldots,m$$

The results are reported in Fig. 40. We can see that the value $\alpha = 1/m$ is the one that maximizes the processing power. This result was expected, since high values of α imply that one memory is the bottleneck of the system, whereas low values of α mean that the accesses are mainly directed to three memories. Both situations increase memory contention and thus decrease system throughput.

The increase in efficiency gained by varying the the number of busses was also analyzed. Fig. 41 shows simulation results for a 6-processor, 4-memory system using a number of busses varying from 1 to 4. The increase in processing power is negligible for low values of p, but becomes very significant for heavily loaded systems. In the latter case the increase in performance clearly shows a "diminishing return" behaviour.

Finally, a 16-processor, 8-memory, 3-bus system was simulated, in order to test the accuracy of the approximate models for large system size. Results are shown in fig. 42. The approximate Markov chains of models A1 and C2, having 46 and 17 states respectively, were solved. The results show that the approximate models behave very well for a system of this size; indeed, the approximate results are so close to the simulation results that they fall within

p	.01	.1	.25	.5	.75	.9	.99
.001	5.994	5.994	5.994	5.994	5.994	5.994	5.994
.01	5.939	5.940	5.940	5.939	5.938	5.937	5.937
.1	5.367	5.379	5.388	5.368	5.268	5.219	5.165
.333	3.903	3.966	4.001	3.818	3.365	3.119	2.823
.5	3.103	3.151	3.178	3.014	2.488	2.186	1.977
.75	2.311	2.313	2.358	2.172	1.704	1.461	1.351
1.	1.787	1.829	1.874	1.699	1.312	1.093	0.995
3.	0.633	0.636	0.641	0.584	0.441	0.378	0.339
5.	0.380	0.384	0.390	0.354	0.262	0.224	0.205

Fig. 40 - Processing power of a 6x4x2 system with non uniform memory reference (values of α in the top row).

p	6x4x1	6x4x2	6x4x3	6x4x4
.001	5.99	5.99	5.99	5.99
.01	5.94	5.94	5.94	5.94
.1	5.16	5.39	5.39	5.39
.3	2.9	4.00	4.11	4.12
.5	1.98	3.18	3.35	3.41
1.	1.02	1.87	2.16	2.16
3.	0.33	0.64	0.81	0.83
5.	0.21	0.40	0.50	0.50
10.	0.10	0.19	0.25	0.26

Fig. 41 - Processing power of a 6-processor, 4-memory system when the number of busses is varied.

p	simulation	A1	C2
.001	15.98	15.98	15.98
.01	15.84	15.84	15.83
.1	14.24	14.27	13.89
.333	8.59	8.73	8.20
.5	6.01	5.99	5.79
1.	2.99	2.99	2.97
3.	1.01	1.00	0.99
5.	0.60	0.60	0.60
10.	0.30	0.30	0.30

Fig. 42 - Simulation and approximate analytic results for a 16x8x3 system.

the simulation confidence interval. Moreover, since the system of linear equations associated with the approximate Markov chain can be easily solved with numerical methods, the approximate models require much less computer time than a simulation program.

APPENDIX 1

In this appendix we give expressions for the transition rates of the approximate Markov chain of model A1 in the general case of a pxmxb system.

Consider that, given that we are in state (i,j), transitions can occur to at most four neighboring states:

$$(i+1,j) \quad (i-1,j) \quad (i,j+1) \quad (i,j-1) \qquad \text{(A1.1)}$$

and we denote such transitions, respectively, with the notation

$$i \dashrightarrow i+1 \quad i \dashrightarrow i-1 \quad j \dashrightarrow j+1 \quad j \dashrightarrow j-1 \qquad \text{(A1.2)}$$

Using the simplifications introduced we associate to these transitions the following rates:

$$R(i \to i+1) = (p-i-j)\,\lambda\,\frac{m-i}{m} \qquad \begin{matrix} 0 \le i < b \\ p-i-j > 0 \end{matrix} \qquad \text{(A1.3)}$$

$$R(i \to i-1) = \left| \begin{matrix} i\,\mu \left|\frac{i-1}{i}\right|^{j} & i<b \\ b\,\mu \left|\frac{b-1}{m}\right|^{j} & i=b \end{matrix} \right. \qquad \text{(A1.4)}$$

$$R(j \to j+1) = \left| \begin{matrix} (p-i-j)\,\frac{i}{m}\,\lambda & i<b, \ p-i-j>0 \\ (p-b-j)\,\lambda & i=b, \ p-b-j>0 \end{matrix} \right. \qquad \text{(A1.5)}$$

$$R(j \rightarrow j\text{-}1) = \begin{cases} i\,\mu \left[1-\left(\frac{i-1}{i}\right)^{j} \right] & i<b \\ b\,\mu \left[1-\left(\frac{b-1}{m}\right)^{j} \right] & i=b \end{cases} \qquad \text{(A1.6)}$$

APPENDIX 2

In this appendix we give expressions for the number of states at level 1 of the exact lumped chain in the case of a pxmx2 system.

We want to count the number of states that show some properties in order to evaluate the transition rates of the approximate Markov models using the averaging technique introduced in section 7.

The level of a state is defined as the difference between the total number of processors and the number of active processors.

At levels 0 and 1 there is only one state, at level 2 there are two states, one with one processor accessing common memory and one with two.

For $1 \geq 3$ we know that we have one state with $n_m=1$ (see eq. 15), but we do not know how many states exist with $n_m=2$. The number of such states can be evaluated by applying some results in combinatorial analysis.

Define the numbers $p_k(n)$ by the recurrent relation

$$p_k(n) = p_k(n-k) + p_{k-1}(n-k) + \ldots + p_1(n-k) + p_o(n-k) \qquad \text{(A2.1)}$$

with

$$p_k(n) = 0 \qquad n<k \;,\; k<0$$

$$p_o(n) = 0 \qquad n>0 \qquad \text{(A2.2)}$$

$$p_k(k) = 1 \qquad k \geq 0$$

Note that $p_k(n)$ is the number of unordered partitions of n into k parts, with k and n integers.

Now we can state that the number of states at level $l=K+2$, $l \geq 3$, such that $n_m=2$ in a pxmx2 system is:

$$n(m,2,k) = \sum_{j=0}^{k} \left| p_2(j+2)\ p_{m-2}(k-j+m-2) \right| \quad , \ k \leq p-2 \qquad \text{(A2.3)}$$

Out of this number, some states will be such that no processor is queueing for a bus to reach a free memory. The number of these states is:

$$n_o(m,2,k) = p_2(k+2) \quad , \ k \leq p-2 \qquad \text{(A2.4)}$$

On the contrary the number of states such that some processor is queueing for a bus is:

$$n_1(m,2,k) = \sum_{j=0}^{k-1} \left| p_2(j+2)\ p_{m-2}(k-j+m-2) \right| \quad , \ k \leq p-2 \qquad \text{(A2.5)}$$

Finally, the number of states at level $l=k+2$, $l \geq 3$ with some processor queueing for a bus (if more than one then all processors queueing for the same memory module) and at least one queue for the busy memories empty, is:

$$n_1'(m,2,k) = \sum_{j=0}^{k-1} p_1(j+1) = k \quad , \ k \leq p-2 \qquad \text{(A2.6)}$$

In the particular case of three memories (m=3) the above results can be put in polynomial form:

$$n(3,2,k) = \left| \begin{array}{ll} \frac{k^2}{4} + k + \frac{3}{4} & \text{K odd} \\ \frac{k^2}{4} + k + 1 & \text{k even} \end{array} \right. \qquad \text{(A2.7)}$$

$$n_o(3,2,k) = \left| \begin{array}{ll} \frac{k+1}{2} & \text{K odd} \\ \frac{k}{2} + 1 & \text{k even} \end{array} \right. \tag{A2.8}$$

$$n_1(3,2,k) = \left| \begin{array}{ll} \frac{k^2}{4} + \frac{k}{2} + \frac{1}{4} & \text{k odd} \\ \frac{k^2}{4} + \frac{k}{2} & \text{k even} \end{array} \right. \tag{A2.9}$$

Acknowledgements

The authors would like to thank Mike Molloy for providing access to some unpublished material and for several helpful discussions, and Mark Hendzel for writing and running the simulation program.

REFERENCES

[AGER79] T. Agerwala "Putting Petri nets to work". IEEE Computer, December 1979, pp.85-94.

[AJMO80] M. Ajmone Marsan and F. Gregoretti "Memory interference models for a multimicroprocessor system with a shared bus and a single external common memory". Submitted to EUROMICRO Journal.

[BASK75] F. Baskett, K.M. Chandy, R.R. Muntz and J. Palacios "Open, closed and mixed networks with different classes of customers". ACM Journal, April 1975, pp.284-260.

[BASK76] F. Baskett and A.J. Smith "Interference in multiprocessor computer systems with interleaved memory". Communications of the ACM, June 1976, pp. 327-334.

[BHAN75] D.P. Bhandarkar "Analysys of memory interference in multiprocessors". IEEE Transactions on Computers, September 1975, pp. 897-908.

[CHAN78] K.M. Chandy and C.H. Sauer "Approximate methods for analyzing queueing network models of computer systems". ACM Computing Surveys, September 1978, pp. 281-317.

[FUNG79] F. Fung and H. Torng "On the analysis of memory conflicts and bus contentions in a multiple-microprocessor system". IEEE Transactions on Computers, January 1979, pp. 28-37.

[HOEN77] S. Hoener and W. Roeder "Efficiency of a multiprocessor system with time-shared busses". EUROMICRO Newsletter, 1977, pp. 35-42.

[HOLT68] A. W. Holt, H. Saint, R. M. Shapiro and S. Warshall "Final report of the information system theory project". Technical report RADC-TR-68-305, Rome Air Development Center, Griffis Air Force Base, New York, September 1968.

[HOLT70] A. W. Holt and F. Commoner "Events and condition". Applied Data Research, New York, 1970.

[HOOG77] C.H. Hoogendoorn "A general model for memory interference in multiprocessors". IEEE Transactions on Computers, October 1977, pp. 998-1005.

[KELL76a] R. M. Keller "Formal verification of parallel programs". Communications of the ACM, July 1976, pp.371-384.

[KELL76b] T.W. Keller "Computer system models with passive resources". PhD Thesis, University of Texas at Austin, 1976.

[KEME60] J.G. Kemeni and J.L. Snell "Finite Markov chains". Van Nostrand, Princeton, 1960.

[MOLL80] M. K. Molloy. Unpublished work.

[NOE71] J. D. Noe "A Petri net model of the CDC 6400". Proceedings of the ACM SIGOPS workshop on performance evaluation, 1971, pp.362-378.

[NOE73] J. D. Noe and G. J. Nutt "Macro E-nets for representation of parallel systems". IEEE Transactions on Computers, August 1973, pp.718-727.

[NUTT72] G. J. Nutt "Evaluation nets for computer systems performance analysis". 1972 FJCC, Montvale, N.J., pp.279-286.

[PETE77] J. L. Peterson "Petri nets". ACM Computing Surveys, September 1977, pp. 223-252.

[PETR66] C. A. Petri "Communications with automata". PhD thesis, translated by C. F. Green, Information systems theory project, Applied Data Research Inc., Princetor.. N.J., 1966.

[PETR73] C. A. Petri "Concepts of net theory". Symposium on mathematical foundations of computer science, High Tatras, September 1973.

[SETH77] A.S. Sethi and N. Deo "Interference in multiprocessor systems with localized memory access probabilities". IEEE Transactions on Computers, February 1979, pp. 157-173.

[SHAP79] S. D. Shapiro "A stochastic Petri net with applications to modelling occupancy times for concurrent task systems". Networks, Winter 1979, pp. 375-379.

[WILL78] P.J. Willis "Derivation and comparison of multiprocessor contention measures". IEE Journal on Computers and Digital Techniques, August 1978, pp. 93-98.

[WULF72] W.A. Wulf and G.C. Bell "C.mmp, a multiminiprocessor". Proceedings AFIPS 1972 Fall Joint Computer Conference.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
UCLA-ENG-8203

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (*and Subtitle*)
MARKOV MODELS FOR
MULTIPLE BUS MULTIPROCESSOR SYSTEMS

5. TYPE OF REPORT & PERIOD COVERED
FINAL

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(*s*)
MARCO AJMONE MARSAN
AND
MARIO GERLA

8. CONTRACT OR GRANT NUMBER(*s*)
N00014-79-C-0866

9. PERFORMING ORGANIZATION NAME AND ADDRESS
University of California Los Angeles
Computer Science Department
Los Angeles, California 90024

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
NRSRO-008

11. CONTROLLING OFFICE NAME AND ADDRESS
Office of Naval Research
1030 East Green Street
Pasadena, California 91106

12. REPORT DATE
February 1981

13. NUMBER OF PAGES

14. MONITORING AGENCY NAME & ADDRESS(*if different from Controlling Office*)
Office of Naval Research
1030 East Green Street
Pasadena, California 91106

15. SECURITY CLASS. (*of this report*)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (*of this Report*)
1. Defense Documentation Center, VA
2. Naval Research Laboratory, D.C.
3. Dr. Slafkosky-Scientific Advisor, D.C.
4. Mr. Gleissner-Naval Ship Research and Development Computation & Mathmatics Dept., MD
5. Captain Grace Hopper (008) Naval Data Automation Command Washington, D.C.
6. Offices of Naval Research in VA, MA, IL, CA, and D.C.

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)
Multiprocessors, busses, interconnection, performance analysis, Markovian models, processing power.

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)
Markovian models are developed for the performance analysis of multiprocessor systems intercommunicating via a set of busses. The performance index is the average number of active processors, called processing power. From processing power a variety of other performance measures can be derived as dictated by the specific processor application. Exact models are first introduced, and are illustrated with a simple example. The computational complexity of the exact models is shown to increase very rapidly with system size, thus making the exact analysis impractical even for medium size systems. To overcome the complexity of computation,

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

several approximate models are introduced. The approximate results are compared with the exact ones and found to be surprisingly accurate for a wide range of configurations. Simulation is used to validate the analytic models and to test their robustness.

END

DATE

FILMED

7-81

DTIC

END

DATE

FILMED

7-81

DTIC